ULBRA CENTRO UNIVERSITÁRIO LUTERANO DE PALMAS

COMUNIDADE EVANGÉLICA LUTERANA "SÃO PAULO"
Recredenciado pela Portaria Ministerial nº 3.607 - D.O.U. nº 202 de 20/10/2005

Ricardo Cardoso de Almeida

DESENVOLVIMENTO DE UM PROTÓTIPO DE FERRAMENTA COLABORATIVA PARA CRIAÇÃO DE RELATÓRIOS A PARTIR DE DIVERSAS ESTRUTURAS DE DADOS

Palmas-TO

2012

Ricardo Cardoso de Almeida

DESENVOLVIMENTO DE UM PROTÓTIPO DE FERRAMENTA COLABORATIVA PARA CRIAÇÃO DE RELATÓRIOS A PARTIR DE DIVERSAS ESTRUTURAS DE DADOS

Trabalho apresentado como requisito parcial da disciplina Trabalho de Conclusão de Curso (TCC) do curso de Sistemas de Informação, orientado pela Professora Mestre Parcilene Fernandes de Brito.

Palmas-TO

2012

Ricardo Cardoso de Almeida

DESENVOLVIMENTO DE UM PROTÓTIPO DE FERRAMENTA COLABORATIVA PARA CRIAÇÃO DE RELATÓRIOS A PARTIR DE DIVERSAS ESTRUTURAS DE DADOS

Trabalho apresentado como requisito parcial da disciplina Trabalho de Conclusão de Curso (TCC) do curso de Sistemas de Informação, orientado pela Professora Mestre Parcilene Fernandes de Brito.

Aprovado em 22 de junho de 2012.

BANCA EXAMINADORA

_____

M.Sc. Parcilene Fernandes de Brito

Centro Universitário Luterano de Palmas

_____

M.Sc. Fernando Luiz de Oliveira

Centro Universitário Luterano de Palmas

_____

M.Sc. Madianita Bogo Marioti

Centro Universitário Luterano de Palmas

Palmas-TO

2012

AGRADECIMENTOS

Primeiro a Deus, pois sem ele nada é possível.

À minha família que teve papel fundamental em toda minha formação. Primeiramente aos meus pais, que definiram quem sou hoje, me deram a maior herança que podia querer, que foi educação, e não foi preciso estudar em escolas particulares para isso, o que aprendi dentro de casa levo para toda minha vida. À minha irmã, Paola, agradeço por ser um grande exemplo na minha vida, sem a qual não teria os sonhos que tenho hoje. À minha irmã, Luana, agradeço por ser uma parceira para todas as horas, trouxe para mim um dos maiores tesouros que eu poderia ser agraciado, minha sobrinha Sophia. À minha prima, Gabriela, que esteve presente na minha vida desde a minha primeira lembrança, antes de prima uma amiga, em que nem a distância pôde abalar e que nos últimos anos nos unimos mais ainda. Por fim, agradeço a todos os familiares que de alguma forma se fizeram presentes nessa minha trajetória.

Agradeço a todos os professores do curso, farei em ordem alfabética para manter a paz (hehe). Ao Andrés, por ser o melhor professor de matemática que pude conhecer, além de suas piadas sem hora marcada. A Cristina, por mostrar a força da disciplina, para dominar um assunto basta disciplina. Ao Edeilson, que a pesar de não ter lecionado disciplinas para mim, é um grande profissional, além de divertido. Ao Fabiano Fagundes, pelas ilusões e histórias contadas no início do curso, isso me estimulou a continuar e lutar, agradeço também ao Fabiano Fagundes pelos desafios lançados, mesmo que indiretamente, pois me fizeram mais forte e seguro para dizer que superei todos com excelência. Ao Fernando, pelas conversas de corredor e por acreditar tanto em mim (principalmente nas inúmeras orientações de Estágio). Ao Jackson, por me permitir conhecer alguém com tamanha genialidade, que por muitas vezes se perde ao tentar aplicar lições de teor humano, mas sempre com as melhores das intenções. A Madianita, a professora mais querida da Universidade, sempre ajudando os alunos e exercendo com excelência o ofício que Deus lhe deu como dom. Ao Michael Shuenck, que teve uma passagem breve no curso, mas se mostrou como um profissional a ser seguido como exemplo, além de ser uma ótima pessoa. Ao Ricardo Marx, que mesmo com seu jeito reservado, nos tirava muitas gargalhadas com suas piadas sem graça na sala de aula.

À Parcilene. Para agradecê-la preciso de um parágrafo inteiro e ainda me faltam palavras. Mais que uma orientadora, se tornou uma amiga. Me ensinou que determinamos onde queremos chegar, que capacidade sem coragem é em vão, me ensinou a aprender a aprender, me mostrou que tudo tem um caminho, que temos que ter um objetivo maior para podermos planejar o caminho e executá-lo. Poderia usar páginas e páginas para expressar

minha gratidão. Agradeço novamente a Deus, por ter tido a oportunidade de conhecer alguém como ela, que além de um exemplo incrível de profissional tem um coração como poucos. Espero ter oportunidade para que nossa história ainda se cruze no decorrer da vida.

À Cleydiane, que não participou das minhas turmas de sala de aula, mas sempre esteve presente durante o curso, que no último um ano se tornou mais que especial e que hoje planejo sonhos maiores com ela. Sem ela não estaria formando, foi mais que força, mais que motivação, uma insistência incansável até em momentos que não queria ouvir nada, tudo fez a diferença. Não sei o que o destino nos reserva, mas é uma pessoa a se ter por perto por toda vida, independente de que papel ela terá ao meu lado, acima de tudo é uma amiga cujo amor já foi cativado e como diria a raposa "tu és eternamente responsável por aquilo que cativas".

Agradeço a todos meus amigos. Tanto os de dentro quanto os de fora da faculdade. Foram muitas as experiências vividas. Agradeço por existirem, mesmo que diversas vezes muitos não entendessem por que tanta dificuldade em formar, ou porque se exigir tanto, mas devo admitir que tudo valeu a pena. Agradeço ao Renan (distante ou não sempre será um grande amigo), ao Flávio (no início do curso tivemos uma série de conflitos para hoje ele ser como um irmão), ao Warley (sempre me deu palavras de incentivo e acreditou em mim), ao Joabe (sabe brincar quando é pra brincar e falar sério quando necessário), a Rayssa (sempre se manteve firme na crença pela minha capacidade e proferiu nesses anos todos palavras de incentivo), ao Heitor e Wisley (pelas diversas histórias que criamos com o copo na mão, até hoje nos divertimos com elas), ao João Neto, Erika, Leandro e Amanda (que mesmo distante de vocês sempre que nos encontramos é como se não tivéssemos sem nos falar).

À todos que de alguma forma fizeram parte dessa trajetória, seja no meio acadêmico ou profissional. Nessa reta final destaco meu agradecimento ao Fábio Castro que cedeu suas férias para que eu pudesse terminar o trabalho de conclusão e ao meu atual chefe/amigo Fernando Martini que me apoiou em um momento tão difícil e fez o possível para que meu emprego não me impedisse de realizar meu trabalho de conclusão.

Agradeço aos meus cachorros, que tiveram que passar sem a atenção do dono que praticamente não pisava mais em casa nessa reta final.

Agradeço ao Gleisson Martins (*in memoriam*), foi um amigo que fiz no curso e que faz falta, com suas brincadeiras e grande coração que sempre que possível ajudava os outros.

SUMÁRIO

RESUMO

Muitas técnicas e ferramentas estão disponíveis no mercado para o tratamento e manipulação de dados, por exemplo, *data warehouse*, *data mining*, ferramentas *OLAP* entre outras. Porém, em muitos casos o gestor necessita de uma análise simples e funcional, que seria rapidamente sanada se ele tivesse uma forma de manipular os dados da base sem a necessidade de um conhecimento técnico específico. Assim, o objetivo do presente trabalho é desenvolver o protótipo de uma ferramenta que a partir de soluções colaborativas baseadas no Modelo 3C de Colaboração sirva de apoio à criação de relatórios, e que ofereça flexibilidade no que tange a possibilidade do usuário importar diversas estruturas de dados para trabalhar. Pelo fato da ferramenta oferecer um ambiente de cooperação que permite que operações efetuadas por um usuário sejam refletidas para os demais usuários que estão trabalhando no mesmo ambiente, é apresentado como solução o padrão de projeto *observer* para gerenciar as dependências entre objetos das diversas estações de trabalho. Além disso, também é apresentada a classe de banco de dados NOSQL como solução para viabilizar a flexibilidade da ferramenta de aceitar diversas estruturas de dados, oferecer alta escalabilidade e disponibilidade, que são características importantes considerando que não se tem informação sobre o volume de dados que o usuário da ferramenta importará.

PALAVRAS-CHAVE: Modelo 3C, NOSQL, Padrão de Projeto OBSERVER

LISTA DE TABELAS

LISTA DE FIGURAS

1 INTRODUÇÃO

O tratamento e análise de dados são peças importantes para o mercado. Frequentemente, gestores buscam soluções em tecnologias para o tratamento e para a manipulação de dados extraídos do ambiente corporativo. Muitas técnicas e ferramentas são aplicadas para este fim como, por exemplo, *data warehouse, data mining*, ferramentas *OLAP*, entre outras. Porém, há casos em que o gestor necessita de acompanhamentos mais simples de dados, que levem em consideração a base de dados operacional da companhia, por exemplo, uma base com informações de um sistema de controle de estoque e vendas que envolva a logística de distribuição de várias filiais de uma empresa.

Geralmente quando se trata de relatórios gerados a partir de bases de dados operacionais, o gestor necessita de mão-de-obra especializada, que entenda suas necessidades e gere relatórios utilizando recursos como linguagem de programação ou consultas diretas a base de dados (através de *scripts*, por exemplo). O desenvolvimento e entendimento da necessidade do gestor por parte do especialista demandam tempo e custo. Muitas vezes, o gestor necessita de uma análise simples e funcional, que seria rapidamente sanada se ele tivesse uma forma de manipular os dados da base sem a necessidade de um conhecimento técnico específico.

Para que uma organização tenha crescimento a partir do tratamento e análise de dados, é necessário que participantes da organização somem esforços para realização dos trabalhos. A constante evolução da interligação de sistemas computacionais em rede permite que hoje seja possível que vários indivíduos com objetivos em comum possam trabalhar sobre um mesmo projeto da organização, isto ocorre em casos de equipes separadas e até mesmo grandes organizações que tem representantes em espaços geográficos distintos. O trabalho em grupo é um ganho para organização, no contexto de análise dos dados, torna possível que se gere novos conhecimentos e memória de aprendizado organizacional. Sistemas Colaborativos possibilitam que essa interação entre indivíduos trabalhando sob um objetivo comum seja realizado via sistemas computacionais.

Segundo Pimentel & Fuks (2011, p. 26), “para que um trabalho seja caracterizado como colaboração, é preciso ocorrer comunicação, coordenação e cooperação”. Ellis, Gibbs e Rein (1991) foram os precursores dessa divisão do trabalho colaborativo em três dimensões,

de modo que, posteriormente, essa classificação deu origem ao Modelo 3C, que aborda de forma detalhada cada dimensão de um ambiente colaborativo. Aplicando o Modelo 3C em Sistemas Colaborativos, todas as três dimensões são gerenciadas através de elementos de percepção. Para exemplificar um possível elemento de percepção é apresentado o seguinte cenário: dentro de um ambiente compartilhado para criação de relatórios, a ferramenta deve informar as operações realizadas por cada indivíduo em tempo real, por exemplo, bloqueando um gráfico que um indivíduo está editando, impedindo que mais de uma pessoa trabalhe sobre um mesmo objeto, evitando problemas de retrabalho e conflitos dentro da equipe.

Essa interação constante entre os usuários no mesmo ambiente compartilhado, através dos elementos de percepção, exige que a concepção da ferramenta seja realizada de forma particular. No caso dos elementos de percepção no cenário de uma ferramenta para criação de relatórios, a operação realizada em um objeto da área compartilhada deve ser informada instantaneamente para os outros objetos contidos na mesma área compartilhada. Para atender esta necessidade do projeto pode-se utilizar o Padrão de Projeto *Observer*, que tem objetivo de "manter consistente as múltiplas visões de um objeto da aplicação, podendo ser aplicado também para manter consistente qualquer conjunto de objetos que apresentem dependências mútuas" (FREITAS, 2003, p. 65).

Sistemas Colaborativos, segundo Assis (2000), devem apresentar preocupações como atrasos na comunicação e sobrecarga dos sistemas. Caso a organização necessite analisar cargas de dados de crescimento escalar, a ferramenta deve prever uma solução para isto. Como solução para este tipo de ambiente pode-se utilizar base de dados NOSQL. O NOSQL atende "requisitos de alta escalabilidade necessários para gerenciar grandes quantidades de dados, bem como para garantir a alta disponibilidade dos mesmos, característica fundamental para as aplicações da Web 2.0" (LÓSCIO et al., 2011, p. 3).

O presente trabalho tem como objetivo desenvolver um protótipo de ferramenta, que tem por finalidade permitir que através de soluções colaborativas a ferramenta sirva de apoio à geração de relatórios, disponibilizando recursos para coordenação, comunicação e cooperação, além de oferecer flexibilidade no que tange a possibilidade do usuário indicar diversas estruturas de dados para trabalhar.

O protótipo terá como escopo a criação de alguns dos possíveis elementos de um relatório (como gráficos e tabelas) em um ambiente gráfico de fácil compreensão que não exija conhecimentos técnicos em TI dos usuários que forem utilizá-la (como lógica de programação, linguagem de programação, linguagem de consultas estruturadas). Além disso, soluções para o desenvolvimento da ferramenta serão abordadas de forma detalhada, são elas:

a utilização do Padrão de Projeto *Observer*, para gerenciar os elementos de percepção da ferramenta (exemplificando isto no protótipo com alertas e controles de edição nos elementos implementados); e o NOSQL, para servir como base de dados que suporta o armazenamento de estruturas de dados variadas e para que não seja necessário que a base importada siga uma estrutura pré-definida, além de solucionar problemas como escalabilidade da ferramenta (por pode trabalhar com grandes cargas de dados sem que afete o desempenho da ferramenta).

O trabalho foi estruturado da seguinte forma: Referencial Teórico, Materiais e Métodos, Resultados e Discussão e Conclusão. Primeiramente, na seção de Referencial Teórico são apresentados os conceitos necessários para a criação do protótipo. Em seguida, na seção de Materiais e Métodos são descritas as ferramentas necessárias para o desenvolvimento do projeto, bem como as tecnologias empregadas. Logo após, na seção de Resultados e Discussão são relatados os passos para a construção do protótipo responsável pela criação de relatórios customizáveis a partir de uma ferramenta gráfica. Finalmente, nas Considerações Finais são apresentados os resultados obtidos com o desenvolvimento do trabalho e possíveis trabalhos futuros.

2 REFERENCIAL TEÓRICO

Esta seção apresenta os conceitos necessários para que seja possível o desenvolvimento do protótipo de ferramenta proposta no presente trabalho. Assim, na seção 2.1 é feita uma descrição sobre sistemas colaborativos construídos a partir do Modelo 3C de Colaboração. Em seguida, é apresentada na seção 2.2 o Padrão de Projeto *Observer* que tem um papel importante na implementação da ferramenta, por focar características de Percepção, tão importantes no Modelo 3C de Colaboração. Por fim, na sessão 2.3 são abordadas teorias e aplicações das bases de dados NOSQL, com objetivo de definir uma solução NOSQL para solucionar problemas no âmbito de desenvolvimento da ferramenta proposta neste trabalho (como a escalabilidade da ferramenta, por exemplo).

2.1 SISTEMAS COLABORATIVOS

Com o advento da interligação dos computadores em rede e a evolução dos recursos oferecidos para *web*, a forma com que as pessoas trabalham nas organizações foi modificada. A organização, segundo Fuks et al. (2002, p. 3), "que era imposta de cima para baixo no paradigma de comando e controle perde eficácia e é substituída por outra menos hierarquizada e mais participativa, onde predominam a comunicação, a coordenação e a cooperação". A evolução das tecnologias de comunição *web* serve de base para auxiliar os trabalhos em grupo, permitindo até mesmo que grandes empresas, distribuídas em diversos pontos geográficos, mantenham ambientes para trabalhos colaborativos, construindo novos conhecimentos na organização de forma colaborativa.

Os sistemas computacionais criados para auxiliar nas atividades colaborativas são denominados *Computer Supported Cooperative Work* (CSCW). Este termo foi utilizado pela primeira vez em 1984, pelos pesquisadores Irene Greif e Paul M. Cashman (COELHO & NOVAES, 2008). Segundo Pimentel & Fuks (2011, p. 13), "Sistemas Colaborativos é a tradução adotada no Brasil para designar ambos os termos *Groupware* e *Computer Supported Cooperative Work*", em que, ainda segundo Pimentel & Fuks (2011), o termo *groupware* para muitos autores é utilizado para trabalhar especificamente os sistemas computacionais que apoiam o ambiente colaborativo.

Na literatura, além do termo Sistemas Colaborativos existem autores que utilizam o termo Sistemas Cooperativos. A definição destes tipos de sistemas variam muito entre os autores, muitos não fazem diferenciação entre os termos colaborativo e cooperativo, já outros abordam de forma distinta. A seguir são apresentadas algumas definições existentes na literatura:

- "Colaboração é uma maneira de trabalhar em grupo, onde os membros do grupo atuam em conjunto visando o sucesso do projeto, sendo que a falha de um dos participantes normalmente implica na falha do grupo como um todo" (GROSZ, 1996 apud GEROSA, 2006, p. 72).
- "O Trabalho Cooperativo é o trabalho que envolve duas ou mais pessoas trabalhando de forma colaborativa, compartilhando informações, sem barreiras e com sinergia" (TAIT et al., 1997, p. 970).
- "O termo cooperação é mais abrangente com distinções hierárquicas de ajuda mútua, ao passo que na colaboração existe um objetivo comum entre as pessoas que trabalham em conjunto sem uma hierarquia" (NITZKE et al., 1999 apud TORRES et al., 2004, p. 3).
- Enquanto a colaboração o trabalho produzido pelo grupo é a principio superior ao de cada membro individualmente, na cooperação o foco é o espaço compartilhado que apoia o trabalho em grupo (FELIPPO, 2008).
- Segundo Panitz (1996 apud TORRES, 2004, p. 4), "a colaboração é uma filosofia de interação e um estilo de vida pessoal, enquanto que a cooperação é uma estrutura de interação projetada para facilitar a realização de um objetivo ou produto final.".

O presente trabalho utilizará o termo Sistema Colaborativo e será adotada a definição de que o Sistema Colaborativo representa um ambiente computacional que propicia a soma de esforços de um grupo para realização de um objetivo comum a todos do grupo. Segundo Pimentel & Fuks (2011, p. 26), "para que um trabalho seja caracterizado como colaboração, é preciso ocorrer comunicação, coordenação e cooperação", para atender esta necessidade será abordado o Modelo 3C de Colaboração.

Na próxima sessão (2.2) será apresentado de forma detalhada o Modelo 3C de Colaboração.

2.1.1 MODELO 3C DE COLABORAÇÃO

O presente trabalho parte do fundamento que Sistemas Colaborativos são divididos em três dimensões: comunicação, coordenação e colaboração. Essas dimensões foram abordadas pela

primeira vez por Ellis, Gibbs & Rein (1991) e, posteriormente, essa classificação deu origem ao Modelo 3C. Pimentel & Fuks (2011) apresentam uma abordagem diferente de Ellis, Gibbs & Rein (1991), em que, substituem o termo colaboração por cooperação, assim, o termo cooperação designa estritamente a ação de trabalho em grupo, enquanto a colaboração representa a ação de realizar todo o trabalho em conjunto, envolvendo os três pilares do Modelo 3C.

Segundo Ferreira (1986 apud FILIPPO, 2008), individualmente as três dimensões do Modelo 3C apresentam as seguintes definições:

- **Comunicação** – comum + ação, é a ação de tornar comum, de trocar mensagens para que haja entendimento comum das ideias discutidas;
- **Coordenação** – co + ordem + ação, é a ação de dispor segundo certa ordem e método, organizar, arranjar;
- **Cooperação** – co + operar + ação, é a ação de operar simultaneamente;

As dimensões do Modelo 3C são analisadas individualmente, considerando os pontos de auxílio à colaboração. Porém, para que haja colaboração os Cs precisam se relacionar, formando um modelo dinâmico e horizontal. Essa relação pode ser observada na Figura 1.

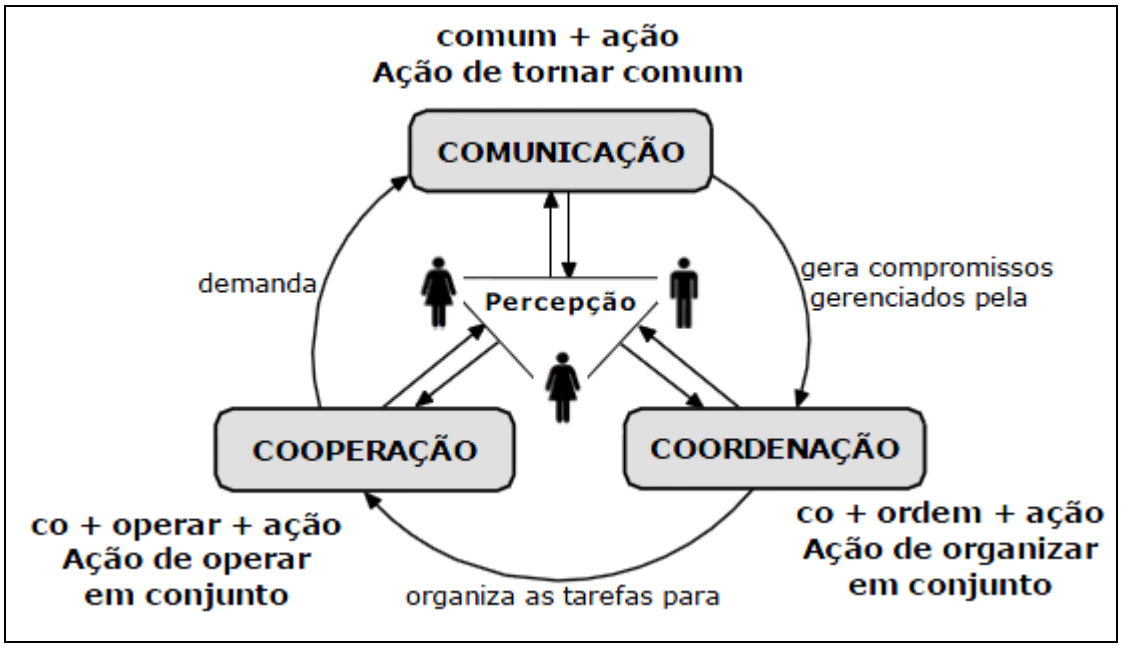


Figura 1 - O diagrama do modelo 3C de colaboração (Gerosa, 2006, p. 77)

A partir da Figura 1 pode-se observar a inter-relação entre os Cs, em que o trabalho em grupo demanda comunicação, que permite a troca de mensagens, objetivando negociações e tomada de decisão, que através da coordenação gera compromissos e distribui tarefas nos

ambientes de cooperação. Para cada dimensão do modelo deve haver mecanismos de percepção, com objetivo de que cada integrante do grupo possa comunicar suas operações e ter um retorno dos outros integrantes, tendo uma visão simultânea de todo trabalho.

As seções a seguir detalham cada dimensão que forma o pilar do Modelo 3C, primeiramente será abordada a dimensão "Comunicação", seu papel no Modelo 3C e sua relação com as outras dimensões do modelo.

2.1.1.1 COMUNICAÇÃO

A comunicação dentro das organizações tem importante papel na troca de mensagens e definição de compromissos. Segundo Fuks et al. (2002), o paradigma de comando e controle, que apresenta uma estrutura com hierarquia bem definida, a comunicação é considerada como bem sucedida quando o emissor tem a confirmação que a mensagem chegou corretamente até o receptor, já quando é aplicada a comunicação sobre o contexto da colaboração o importante é assegurar o entendimento da mensagem, para que a intenção do emissor resulte em compromissos assumidos para o sucesso do trabalho compartilhado.

O ambiente colaborativo deve apresentar as opções de comunicação bem estruturadas e de preferências com formatos diversos. Essas características buscam permitir redução de ruídos no envio da informação. Caso haja um mau entendimento entre o emissor e o receptor da informação, poderá prejudicar a execução dos compromissos firmados e afetar diretamente no trabalho do grupo.

A comunicação mediada por computador trouxe diversas possibilidade para interação social, entre elas Pimentel & Fuks (2011, p. 66) destacam: "assincronicidade, ausência da interação face a face, anonimato, privacidade, contato contínuo com interlocutores sempre conectados *online*, comunidades virtuais, entre outras". Oliveira & Tadesco (2010, p. 19) destaca que "a *Web* 2.0 transforma o antigo modelo da *Web* tradicional, permitindo que os usuários antes passivos da informação sejam produtores e difusores desta". A *web* 2.0 oferece vários recursos que permitem trabalhar de forma mais interativa nos ambientes colaborativos. Diversas ferramentas podem ser utilizadas para minimizar ruídos e intensificar a comunicação do grupo, estas ferramentas, no que se refere, a tempo de resposta podem ser:

- **Assíncronas**: em que o emissor envia uma mensagem e não espera resposta rapidamente, as mensagens ficam registradas e podem ser acessadas a qualquer momento pelo receptor.
- **Síncronas**: em que os elementos de percepção agem em um intervalo de tempo pequeno, quase imediato, fazendo com que as mensagens entre os interlocutores sejam trocadas instantaneamente.

Segundo Fuks (2003 apud BORGES et al., 2007, p. 53), considerando o tipo de aplicação as ferramentas de comunicação podem se dividir em:

- **Sistemas de mensagens**: suportam a troca de mensagens de texto entre usuários.
- **Editores multiusuários**: onde códigos podem ser alterados por vários usuários.
- **Decisão em grupo**: oferecem mecanismos para tomada de decisão em grupo.
- **Conferências**: são em geral módulos com apoio de áudio e vídeo.
- **Agentes Inteligentes**: sistemas de software autônomos.
- **Sistemas de coordenação**: permitem o controle e gerência das atividades do grupo.

A Figura 2 apresenta a estrutura de comunicação dentro de um ambiente colaborativo.

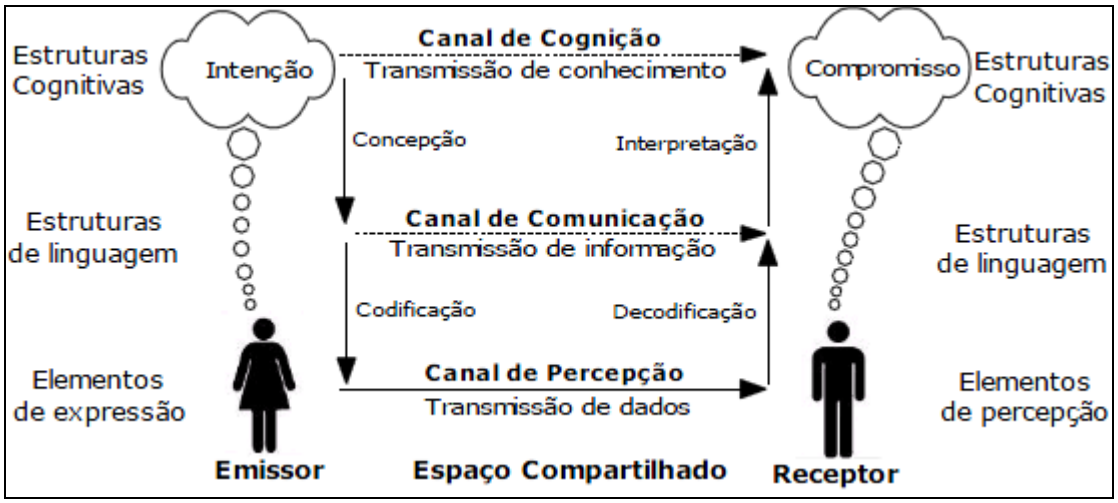


Figura 2 - Modelando a comunicação (FUKS, 2002, p. 4)

Analisando a Figura 2 é possível notar a forma de comunicação entre o emissor e receptor. O emissor, de forma externa ao ambiente colaborativo, formula sua intenção. A intenção do emissor é transcrita por ele utilizando elementos de expressão do ambiente (símbolos, texto, ícones, mensagens pré-definidas, vídeo, áudio etc.). Após a construção da mensagem, o receptor utiliza elementos de percepção para realizar o envio. Os elementos de percepção são definidos "como sendo o conhecimento das atividades dos outros que provê contexto para a sua própria atividade" (DOURISH, 1992 apud FILIPPO, 2008, p. 56), informando, assim, as interações no ambiente colaborativo, mostrando para todo o grupo, por exemplo, que um usuário está atribuindo uma nova atividade. Assim, o receptor através do canal de percepção recebe a mensagem, interpreta e assume os compromissos gerados.

A visão apresentada na Figura 2 é centrada na construção de compromissos, que levam a ações. Segundo Pimentel & Fuks (2011), esse modelo, em que os interlocutores trocam mensagem para negociar e assumir compromissos, é recorrente de um estudo bastante difundido chamado "conversão-para-ação". A Figura 3 apresenta o conjunto de estados que compõe o processo de "conversação-para-ação".

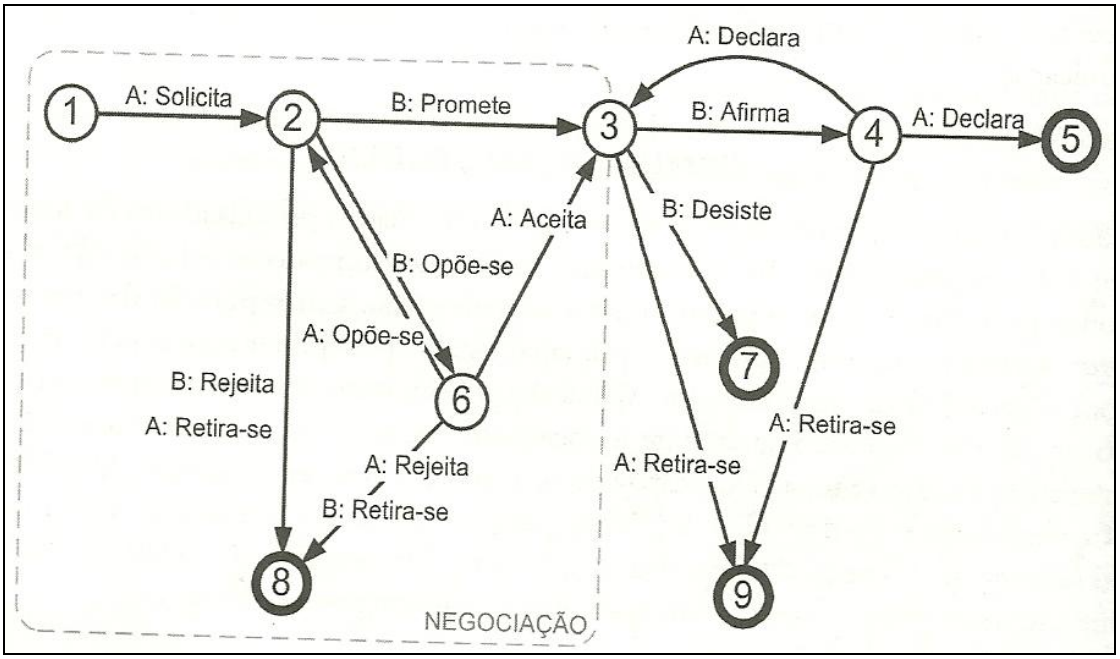


Figura 3- Diagrama da Conversão para Ação (WINOGRAD, 1986, p. 206, imagem adaptada)

Pode-se observar na Figura 3 o fluxo do modelo de conversão-para-ação, em que o emissor A solicita ao receptor B uma determinada ação, o receptor B pode prometer realizar a tarefa ou se opor e argumentar com emissor A, caso B se oponha, A pode renegociar, rejeitar ou aceitar as argumentações de B. Todos os outros pontos do diagrama representam as possíveis interações entre os interlocutores A e B durante todo processo da solicitação inicial de A, regidos por atos de fala que promovem as mudanças.

Para Fuks (2002, p. 5), "para garantir o cumprimento destes compromissos e a realização do trabalho colaborativo através da soma dos trabalhos individuais, é necessária a coordenação das atividades". A seção a seguir abordará a dimensão da Coordenação no Modelo 3C de Colaboração e sua importância para garantir que os esforços de comunicação e cooperação não sejam perdidos.

2.1.1.2 COORDENAÇÃO

O papel da coordenação no Modelo 3C, para Fuks et al. (2008, p. 639, tradução nossa), é tida como "o elo entre os outros dois Cs, a fim de aplicar o sucesso da colaboração". Entende-se por este elo entre as outras duas dimensões, do Modelo 3C, que os compromissos firmados na comunicação devem ser coordenados no processo de cooperação. Gerosa (2006, p. 77) afirma que "as pessoas, as tarefas e os recursos são gerenciados para lidar com conflitos e evitar a perda dos esforços de comunicação e de cooperação", a seguir estão listados os elementos que precisam ser coordenados segundo Fuks et al. (2008):

- **Pessoas** – A coordenação das pessoas está profundamente relacionada com a comunicação e contexto.
- **Recursos** – Representa os objetos que são trabalhados nos ambientes compartilhados, logo está relacionado com a etapa de cooperação.
- **Tarefas** – São os elementos que compõe as atividades colaborativas e estão ligadas por interdependências, segundo Filippo (2008, p. 53), as tarefas "podem ser atômicas ou subdivididas em várias tarefas que, por sua vez, têm interdependências entre si".

Para Schmidt & Bannon (1992 apud FILIPPO, 2008, p. 52), "a coordenação, entendida como sinônimo de articulação, é o conjunto de atividades necessárias para gerenciar a natureza distribuída da colaboração". Considerando essa definição de articulação das tarefas, podem-se dividir as etapas da coordenação em:

- **Pré-articulação** – Esta etapa normalmente ocorre antes que o trabalho colaborativo seja iniciado. Nesta fase é determinada, por exemplo, a identificação dos objetivos, a conversão destes objetivos em tarefas, a seleção dos grupos de trabalho, a distribuição das tarefas, entre outras definições necessárias para organização do trabalho. (FUKS et al., 2002, p. 5);
- **Gerenciamento das tarefas** – Que corresponde à fase dinâmica da articulação. Neste sentido mais restrito, a coordenação é a ação de gerenciar interdependências entre as atividades realizadas para que um objetivo seja alcançado (MALONE, 1994 apud FILIPPO, 2008);
- **Pós-articulação** – Tem por responsabilidade a avaliação e análise das tarefas finalizadas, além da documentação do processo de colaboração (com objetivo de manter a memória do processo) (FUKS et al., 2002, p. 5).

Dentre as etapas citadas anteriormente, o gerenciamento para andamento das tarefas para Fuks et al. (2002, p. 5) "é a etapa mais importante da coordenação, pois é a parte mais

dinâmica da mesma, precisando ser renegociada de maneira quase contínua ao longo de todo o tempo". Nesse gerenciamento, as interdependências entre as tarefas podem ser baseadas no tempo ou recurso.

Quando a interdependência é baseada em tempo, as tarefas de uma atividade são definidas de forma sequencial (FUKS et al., 2008), por exemplo, para um integrante realizar testes de unidade em um componente pertencente a um projeto de desenvolvimento, anteriormente, deve-se finalizar a atividade de implementação do componente em questão. Já quando a interdependência é baseada em recursos, "a coordenação descreve como lidar com o acesso sequencial ou simultâneo de múltiplos participantes a um mesmo conjunto de objetos de cooperação" (FUKS et al., 2002, p. 6). Por exemplo, quando em um editor de texto compartilhado os integrantes trabalham sobre o mesmo parágrafo e o ambiente coordena para que o parágrafo seja bloqueado quando algum integrante estiver editando o mesmo.

A Figura 4 apresenta a estrutura de coordenação dentro de um ambiente colaborativo.

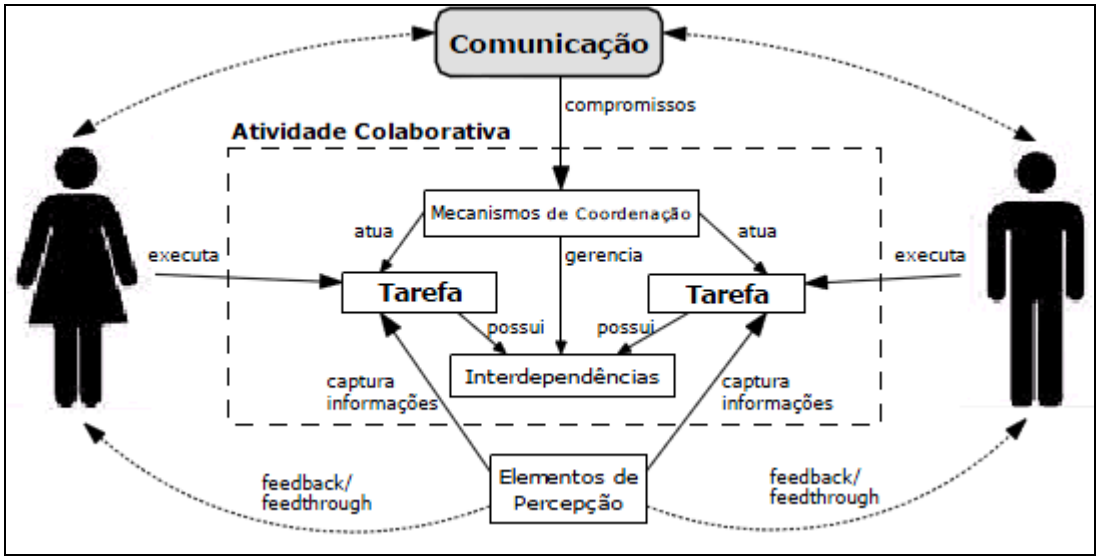


Figura 4 - Modelando a coordenação (FUKS et al., 2002, p. 6)

Ao analisar a Figura 4, pode-se perceber que através da dimensão da comunicação, compromissos são firmados, esses compromissos desencadeiam tarefas para os integrantes do grupo. Assim, a partir dos mecanismos de coordenação as interdependências entre as tarefas são estabelecidas, definindo também os integrantes que executarão cada tarefa. Por fim, através dos elementos de percepção os integrantes tem a visão de toda execução da tarefa, podendo renegociar ou executar as tarefas, em um processo dinâmico. Essa interdependência entre as tarefas podem apresentar um laço forte ou fraco de integração, a intensidade de integração entre as tarefas é descrita a seguir:

- **Fracamente integradas**: são tarefas que não necessitam de mecanismos explícitos de coordenação, em que a coordenação é gerida através do protocolo social, que "é caracterizado pela confiança na habilidade dos participantes de mediar as interações e na ausência de uma coordenação explícita da atividade" (FILIPPO, 2008, p. 54);
- **Fortemente integradas**: são tarefas que "exigem sofisticados mecanismos de coordenação a fim de ser suportados por sistemas de computador" (FUKS et al., 2008, p. 640, tradução nossa). Isto ocorre, por exemplo, no desenvolvimento de um software que as tarefas têm dependências maiores para serem iniciadas, executadas ou finalizadas, nesse caso, lista de tarefas, gerenciador de atividades, *workflow*, são exemplos de mecanismos de coordenação que poderiam ser utilizados.

Segundo Fuks et al. (2008), o maior desafio é montar um ambiente flexível que permita aos integrantes terem mecanismos de coordenação tanto utilizando o protocolo social quanto condições mais rígidas (no caso de apresentar tarefas fortemente integradas). O ambiente colaborativo deve prever o dinamismo de integração entre os integrantes, com objetivo de evitar conflitos. Logo, a total rigidez dos mecanismos de coordenação de um ambiente colaborativo não é aconselhável.

A condução dos trabalhos em grupo segundo Pimentel & Fuks (2011) podem ser conduzidas da seguinte maneira:

- **Individualmente**: cada integrante do grupo conduz sua parte do trabalho, a soma das partes compõe o todo, não havendo dependência entre as partes. Pode-se exemplificar com uma atividade de *brainstorming* em que cada integrante formula suas ideias e dispõe em um ambiente compartilhado;
- **Com repasse de tarefas**: as atividades dos integrantes estão interligadas e há necessidade de trocar ideias e passar tarefas e resultados dentro do grupo. Ocorre, por exemplo, quando uma equipe de gerência de projetos realiza levantamento de requisitos e precisam interagir com a equipe, trocando informações e experiências para que todos tenham uma visão comum do sistema e gerem a documentação ao final;
- **Orquestrando**: as atividades são mais interligadas e há dependência forte entre elas. As atividades individuais precisam ser fortemente coordenadas, pois somente um esforço conjunto sincronizado leva à solução do problema. Um exemplo dessa situação é quando uma equipe de desenvolvimento de *software* executa a implementação de vários módulos e precisa organizar para que a interface, código e estrutura de dados e testes sejam

realizados de forma ordenada, para ao final ter um *software* com interface e codificação padronizados.

Apesar das dimensões de comunicação e coordenação serem muito importantes no ambiente colaborativo, ainda assim não são suficientes. Considerando que a coordenação segundo Raposo et al. (2004 apud FUKS et al., 2008, p. 640, tradução nossa) "é necessária para gerenciar as tarefas, de acordo com o modelo 3C é igualmente necessário prever um espaço de trabalho compartilhado, onde a cooperação terá lugar". A seção a seguir abordará a dimensão da Cooperação no Modelo 3C de Colaboração.

2.1.1.3 COOPERAÇÃO

No trabalho em grupo Gerosa (2006, p. 102) define que "cooperação é a operação conjunta dos participantes no espaço compartilhado, visando a realização das tarefas". Assim, durante o processo de cooperação os integrantes de um grupo podem: produzir, manipular, refinar e organizar objetos, como por exemplo, atas de reuniões, relatórios, mapas conceituais etc. Ainda segundo Gerosa (2006, p. 102), para trabalhar com os objetos compartilhados, "os participantes contam com mecanismos de expressão, e para se informar dos resultados de suas atuações (*feedback*) e das ações de seus colegas (*feedthrough*) dispõem de informações de percepção".

Em relação ao espaço de trabalho, no que se refere à localização dos integrantes, é fundamental que se faça a distinção dos trabalhos realizados no mesmo local e em lugares distantes ou remotos. A Figura 5 apresenta a relação de espaço e tempo das interações.

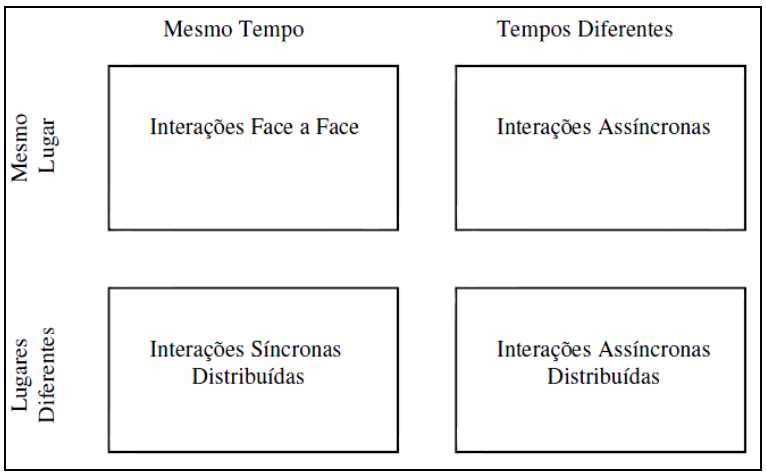


Figura 5 - Matriz de tempo e espaço de ferramentas groupware (ELLIS et al., 1991, p. 41, imagem adaptada)

Ao analisar a Figura 5, pode-se notar que no caso das interações aplicadas com os integrantes compartilhando do mesmo lugar, é possível trabalhar ao mesmo tempo ou em tempos diferentes. Quando a interação nesse caso é ao mesmo tempo as interações ocorrem face a face, por exemplo, no formato de reuniões, já no caso de participação dos integrantes em tempo diferentes as interações são assíncronas, mesmo que estejam fisicamente no mesmo local, podendo, por exemplo, em um cenário hospitalar, as interações ocorrerem via prontuário.

Interações ocorridas em lugares distantes (Figura 5) permitem vislumbrar mais facilmente aplicações para ambientes colaborativos, pois, pelo fato dos integrantes estarem distribuídos fisicamente, cria-se a necessidade de oferecer ambientes compartilhados. Nesse caso a cooperação pode ocorrer de forma síncrona, em que os participantes interagem ao mesmo tempo, por exemplo, em *chats*, conferências, áreas de trabalho compartilhadas etc., ou pode ocorrer de forma assíncrona, em que os integrantes interagem em tempos diferentes, como em fóruns, repositório de arquivos etc. Para evitar conflitos e garantir o sucesso do trabalho, é necessário aplicar corretamente os elementos de percepção.

Como exemplo de aplicação de elementos de percepção pode-se apresentar um cenário em que, em um ambiente com vários integrantes trabalhando sob um mesmo texto, é possível inserir um elemento de percepção para que o ambiente colaborativo indique qual integrante está trabalhando em qual parágrafo do texto, assim todos integrantes teriam a visão do processo de cooperação síncrono e poderiam evitar conflitos e retrabalho. Os elementos de percepção são muito importantes, e estão relacionados com todas as dimensões do modelo. Porém, no que tange a dimensão de cooperação uma atenção especial deve ser dada aos elementos de percepção, pois, eles oferecem informações importantes para execução do trabalho colaborativo, reduzindo interrupções entre os integrantes, já que a informação já está acoplada aos elementos de percepção.

Segundo Fuks et al. (2002, p. 8), “o excesso de informações pode causar sobrecarga e dificultar a colaboração. Para evitar a sobrecarga, é necessário balancear a necessidade de fornecer informações com a de preservar a atenção sobre o trabalho”. Logo, os elementos de percepção aplicados em um sistema colaborativo devem ser muito bem avaliados, quando possível analisar contexto, fatores culturais e necessidades priorizadas. Gerosa (2006, p. 105) analisa a definição dos elementos de percepção considerando os indivíduos, em que, “como cada um tem suas capacidades, necessidades e preferências, os elementos devem ter flexibilidade o suficiente para se adequarem às diferentes personalidades”.

A Figura 6 apresenta a estrutura de cooperação dentro de um ambiente colaborativo.

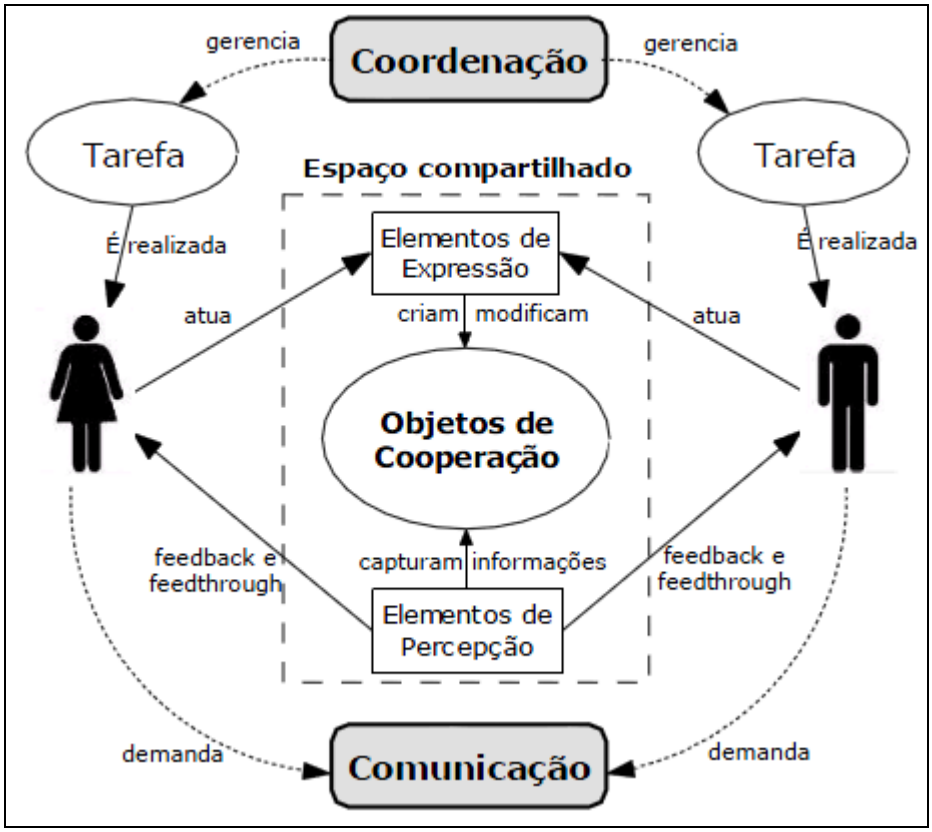


Figura 6 - Modelando a cooperação (FUKS et al., 2002, p. 7)

Analisando a Figura 6 pode-se perceber que através da dimensão da coordenação, as tarefas são distribuídas, os integrantes promovem a cooperação na execução da tarefa em um espaço compartilhado, atuando nos elementos de expressão, para criar ou compartilhar objetos de cooperação. Por fim, através dos elementos de percepção, os integrantes conseguem visualizar o andamento do trabalho e ter informações necessárias para assumir comportamentos, e caso necessário iniciar o processo de comunicação, para troca de ideias, renegociações ou resolução de conflitos.

"O registro das interações dos membros do grupo fica armazenado, catalogado, categorizado e estruturado nos objetos de cooperação" (FUKS et al., 2002, p. 7). Desta forma é possível garantir a memória do grupo nos projetos colaborativos. Conhecimentos construídos informalmente, como em discussões, fatos, pontos de vista etc., são mais difíceis de capturar, entretanto ter este histórico armazenado é interessante, considerando a possibilidade de se obter os motivos que uma decisão foi tomada, validando-as e criando embasamento para tomada de decisão (GEROSA, 2006).

A seção a seguir aborda ferramentas *groupware*, destacando suas características, tipos e ligações com o Modelo 3C de Colaboração.

2.1.2 GROUPWARE

Como já abordado na sessão 2.1 o termo "Sistema Colaborativo" é a tradução adotada no Brasil para abordar tanto os termos *groupware* quando CSCW, em que, para muitos autores o termo *groupware* é utilizado para trabalhar especificamente os sistemas computacionais que apoiam o ambiente colaborativo. Para Ellis, Gibbs e Rein (1991, p. 40, tradução nossa), o objetivo do *groupware* é "dar assistência aos grupos na comunicação, colaboração e coordenação de suas atividades". Essa definição é totalmente convergente com o Modelo 3C de Colaboração, por abordar os três pilares do modelo. Ainda segundo Ellis, Gibbs e Rein (1991, p. 40, tradução nossa), pode-se definir *groupware* como sendo "sistemas de computadores que suportem grupos de pessoas engajadas em uma tarefa (ou meta) e que proveem uma interface para um ambiente compartilhado".

Para Assis (2000), *groupwares* devem ser analisados de forma distinta comparados a sistemas tradicionais, pois abordam o trabalho em grupo e o comportamento das pessoas nesses ambientes, levam em consideração aspectos e entendimento de tecnologias de rede, preocupações como atrasos na comunicação e sobrecarga dos sistemas, por afetar diretamente na experiência do usuário com o ambiente colaborativo. Wilson (1991 apud SICA, 1996, p. 5) sintetiza essa forma de análise indicando que "devido ao envolvimento de pessoas e recursos computacionais, sistemas baseados em CSCW podem ser discutidos sob dois aspectos distintos: aspectos humanos e aspectos tecnológicos". Desta forma é possível visualizar problemáticas tanto no contexto humano (por exemplo, as relações entre os integrantes e o tratamento de conflitos), como tecnológico (por exemplo, um ambiente de muitos usuários sobre o mesmo objeto de colaboração, exigindo técnicas como controle de concorrência).

Sica (1996) apresenta definições dos dois aspectos, que foram sintetizadas e estão listadas a seguir:

- Aspectos humanos
    - **Individuais Humanos** – Tem por objetivo aproximar o sistema computacional às situações reais do ambiente de trabalho físico do usuário. Apesar de o trabalho ser realizado em grupo, sua constituição representa a soma de conteúdos elaborados previamente de forma individual, com suas próprias técnicas e metodologias. O que torna interessante, por exemplo, considerar as preferências pessoais de interface de cada usuário e uso da estação de trabalho de forma momentânea ou exclusiva para cada integrante;

- **Aspecto Organizacional** – Está relacionado com os componentes da organização, no que tange a estrutura, cultura, conhecimento e gerenciamento. Tanto o conhecimento organizacional quanto humano podem ser afetados pela estrutura organizacional que define a área de atuação de cada integrante da instituição. As informações comuns da organização devem estar disponíveis para os membros sempre que necessário. Considerando a aplicação de sistemas colaborativos na instituição, é interessante considerar fatores como, por exemplo, fornecimento de um repositório com conhecimento organizacional e humano e, também, ser possível através do gerenciador dos trabalhos colaborativos, utilizar opções para controle e andamento das atividades, flexível suficiente para permitir a adaptação conforme a necessidade;
- **Ponto de Vista da Dinâmica de Grupo** – Abrange o desenvolvimento do trabalho, formas de tomada de decisão, consulta aos participantes e comportamento do grupo. Para analisar as ferramentas colaborativas necessárias deve-se considerar o tamanho do grupo, cultura dos integrantes, distribuição em tempo e espaço.

- Aspectos tecnológicos
  - **Suporte as atividades de Grupo** – Forma de organização e controle do processo de composição do trabalho colaborativo, estabelecendo previamente as funções e papéis de cada integrante do grupo;
  - **Suporte ao Espaço de Trabalho Compartilhado** – Permite que integrantes de um mesmo trabalho desenvolvam as atividades de forma sincronizada e em um mesmo ambiente de cooperação;
  - **Suporte ao Compartilhamento de Informação** – Possibilita que usuários trabalhem em um mesmo conjunto de informações, para isso é interessante que existam facilitadores, como gerenciadores de documentos e histórico de alterações;
  - **Suporte à Comunicação** – Responsável pela troca de mensagens variando em demandas de tempo e espaço, deve oferecer instrumentos para preparar, trabalhar e compartilhar os dados;
  - **Suporte à Transparência** – Ligado ao tratamento das informações sobre o processo colaborativo, como localização dos usuários, tempo de interação, interfaces distintas etc.. Deve-se tomar cuidado com essas informações, pois fatores que possam atrapalhar o andamento do trabalho devem ser ocultados, desta forma é preciso priorizar a transparência das informações necessárias para o andamento do trabalho.

A análise dos aspectos humanos e tecnológicos torna importante uma abordagem crítica sobre fatores de risco no desenvolvimento de *groupware*. Segundo Grudin (1993 apud ZOTTO 2000), os fracassos no desenvolvimento de componentes de *groupware* são motivados por cinco fatores de risco:

- **A disparidade entre quem faz o trabalho e quem recebe o benefício** – Novas solicitações de componentes de *groupware* são realizadas, entretanto os solicitantes não irão beneficiar-se diretamente com o trabalho solicitado;
- **Fatores sociais, políticos e motivacionais** – O *groupware* pode trabalhar atividades que levantem assuntos sociais polêmicos, mexa com estruturas políticas existentes, ou desmotive usuários importantes para o sucesso da tecnologia;
- **Manipulação de exceções em grupos de trabalho** – O *groupware* deve ser flexível, possibilitando a manipulação de exceções e improvisações, conforme necessidades específicas das atividades de grupo;
- **A subestimada dificuldade em avaliar groupware** – Devido a complexidades dos *groupwares*, são introduzidos obstáculos para compreensão, análise de forma generalizada e avaliação;
- **A quebra da tomada de decisão intuitiva** – As intuições realizadas para o desenvolvimento de *groupware* são geralmente fracas, sendo um fator corrente em aplicativos multiusuários.

Com objetivo de facilitar a especificação dos componentes de *groupware* Tietze (2001) apresenta uma lista de requisitos desejáveis. O autor separa os requisitos em dois grupos, requisitos de usuário (RU) e requisitos de desenvolvedor (RD), respectivamente "(1) o desenvolvedor, quem usa o sistema para desenvolver um aplicativo, e (2) o usuário final, quem usa esta aplicação" (TIETZE, 2001, p. 7, tradução nossa). O autor apresenta sua lista de requisitos através de cenários para exemplificar a aplicação de cada requisito, os cenários apresentados no presente trabalho não se referem aos cenários apresentados por Tietze (2001). Os requisitos são:

- **RU1 – Acesso aos objetos compartilhados e às ferramentas de colaboração –** O "Usuário A" utiliza a ferramenta colaborativa para desenvolver relatórios e necessita que em sua estação de trabalho estejam os projetos compartilhados que ele (Usuário A) tem permissão para colaborar, com as ferramentas colaborativas necessárias para manipular os objetos e interagir com os outros usuários que compartilham dos mesmos projetos;

- **RU2 – Auxílio na escolha das ferramentas apropriadas –** O "Usuário A" necessita desenvolver um relatório de forma colaborativa com o "Usuário B", em que o único meio de contato entre os usuários é de forma remota. Considerando a variedade de ferramentas e possibilidades de como manipulá-las o ambiente auxilia na escolha da ferramenta apropriada para realizar as tarefas. Por exemplo, caso o "Usuário A" esteja editando um objeto compartilhado "Tabela" o ambiente oferecerá ferramentas para definir filtros, configurações de interface e controles de edição para que o as alterações do "Usuário A" sejam informadas para o "Usuário B" e não interfira no andamento do trabalho cooperativo;
- **RU3 – Fornecimento de informações de percepção –** Ao acessar o ambiente cooperativo o "Usuário A" visualiza que o "Usuário B" está conectado e um dos gráficos do relatório está sendo editado por ele (impedindo o "Usuário A" de editá-lo neste momento);
- **RU4 – Colaboração síncrona e assíncrona –** O ambiente fornece serviços de colaboração síncrona e assíncrona, de modo que os participantes selecionam o modo de interação mais adequado a cada situação. Por exemplo, no momento em que um parágrafo do relatório está sendo desenvolvido pelo "Usuário A", surge um questionamento que o "Usuário A" através de uma ferramenta de comunicação síncrona solicita explicação ao "Usuário B";
- **RU5 – Acesso ao ambiente independente da estação de trabalho –** Em certo momento do trabalho o "Usuário B" comunica ao "Usuário A" a necessidade de recolher informações *in loco* do curso que o projeto compartilhado é referente, para isso o "Usuário A" se desloca até a Universidade e recolhe as informações necessárias, por fim, se conecta a um computador dos laboratórios de informática da Universidade, sendo possível continuar o trabalho colaborativo com o "Usuário B" do ponto onde foi interrompido;
- **RU6 – Fornecimento de espaço privativo e público e transição entre eles –** O "Usuário A" quer fazer um parágrafo de introdução para o relatório, porém o "Usuário A" não quer que o "Usuário B" tenha acesso a versão prévia de seu texto. Quando seu parágrafo estiver mais consolidado o "Usuário A" liberará o conteúdo para que o "Usuário B" possa visualizar e trabalhar o parágrafo;
- **RU7 – Extensão dinâmica do ambiente –** Os usuários A e B necessitam criar um novo tipo de gráfico que não consta na ferramenta (um gráfico tridimensional, por exemplo). O "Usuário A" acha no site do fornecedor do sistema um novo pacote que permite a criação do tipo de gráfico que necessitam para o projeto, assim, o "Usuário A" incorpora no ambiente compartilhado, de modo que o "Usuário B" também tenha acesso;

- **RU8 – Sincronização entre ferramentas diferentes –** O "Usuário A" reposiciona os textos e gráficos do relatório, enquanto o "Usuário B" atualiza a configuração de um dos relatórios. As alterações feitas em uma ferramenta são refletidas na outra, desta forma, o "Usuário A" visualizará o novo conteúdo do gráfico editado pelo "Usuário B", a tempo que, o "Usuário B" visualizará o novo posicionamento do gráfico que acabou de editar;
- **RU9 – Mobilidade –** O "Usuário A" está em uma reunião na Universidade apresentando os demonstrativos obtidos em alguns dos relatórios gerados pela ferramenta colaborativa, porém o "Usuário A" é informado que houve alterações em um dos relatórios, no meio da reunião o "Usuário A" acessa o ambiente compartilhado através de seu *Smartphone* (ambiente adaptado devido o tamanho da tela) e se atualiza sobre as modificações do relatório;
- **RU10 – Agrupamento de ferramentas –** O "Usuário A" sempre que trabalha com relatórios sobre demonstrativos de acesso a materiais didáticos utiliza em geral gráficos pizza, texto plano e chat para interagir de forma síncrona com outros integrantes do projeto. Assim, o "Usuário A" agrupa as ferramentas que usa com frequência e quando é colocado em situação de relatórios similares, utiliza esse agrupamento facilitando seu trabalho;
- **RU11 – Alta performance –** Ao trabalhar em um ambiente cooperativo os participantes esperam que o trabalho se dê de forma natural, assim a ferramenta tem que prever questões como atraso devido a latência do sistema. Por exemplo, no momento em que o "Usuário A" passa um gráfico para o modo de edição, de forma quase que instantânea o "Usuário B" deve visualizar essa operação e ser impedido de trabalhar no mesmo gráfico até que o "Usuário A" libere o gráfico para novas edições, evitando possíveis transtornos no trabalho colaborativo;
- **RD1 – Reuso da experiência e conhecimento anteriores –** O desenvolvimento de ferramentas colaborativas envolve diversas tecnologias, além da aplicação de novas tecnologias. Por exemplo, o "Desenvolvedor A" deseja que sua experiência e conhecimentos sobre padrão de projeto *observer* sejam aproveitados;
- **RD2 – Aproveitamento do modelo de dados –** O "Desenvolvedor A" deve reaproveitar os modelos de dados já existentes sempre que possível, caso o "Desenvolvedor A" estiver dando manutenção em um sistema colaborativo para criação de relatórios e vá desenvolver uma nova forma de gráfico, ele irá reaproveitar o modelo de dados utilizados pelos outros gráficos e aplicar para esta nova situação;

- **RD3 – Compartilhamento transparente de objetos de dados –** O "Desenvolvedor A" não deve se preocupar com a distribuição e o compartilhamento dos objetos de dados do sistema, a ferramenta de desenvolvimento deve oferecer esse suporte;
- **RD4 – Suporte a objeto de dados locais e compartilhados –** O "Desenvolvedor A" deve decidir se seus objetos de dados serão compartilhados ou se serão apenas estruturas temporárias alocadas localmente. Independente da escolha do "Desenvolvedor A" a infraestrutura deve dar suporte as duas situações;
- **RD5 – Acesso à informações de cooperação quando necessário –** Caso o "Desenvolvedor A" queira melhorar a ferramenta colaborativa de edição de texto para relatórios, ele precisa ter acesso a todas as informações sobre a configuração atual de colaboração. Essa prática deve oferecer elementos de percepção para que o "Desenvolvedor A" esteja ciente das atividades e tarefas dos outros usuários, apresentando, por exemplo, quem já trabalhou sob o mesmo artefato, quem está trabalhando e/ou como está trabalhando, e até mesmo apresentando interações futuras como quem vai trabalhar com um determinado artefato;
- **RD6 – Disponibilização de novas ferramentas –** O "Desenvolvedor A" criou uma nova ferramenta para integrar o sistema colaborativo para elaboração de relatórios, agora os usuários poderão incluir rodapé configurável nos relatórios. A infraestrutura não deve oferecer dificuldades para que o "Desenvolvedor A" implante essa nova ferramenta, e todos usuários finais devem receber a atualização no mesmo momento, evitando problemas com o trabalho colaborativo da equipe. Vale ressaltar que este requisito deve ser atendido tanto para novas ferramentas quando para atualizações de ferramentas existentes;
- **RD7 – Escalabilidade –** A performance da infraestrutura deve apresentar uma degradação suave do desempenho conforme novos usuários se conectem. Já recursos desenvolvidos na ferramenta que trabalhem com grandes quantidades de dados, devem ser tratados sob responsabilidade dos desenvolvedores, por exemplo, a carga de grandes quantidades de dados para elaboração de relatórios, o "Desenvolvedor A" deve se preocupar em não fazer essa carga e manipulação de dados afetar no desempenho do trabalho colaborativo dos usuários finais;
- **RD8 – Suporte a integração com arquiteturas externas –** Os desenvolvedores necessitam de uma interface aberta para colaboração, permitindo consultar o estado atual da colaboração, invocar ferramentas colaborativas externas, etc;

- **RD9 – Suporte a ferramentas localizadas no servidor –** A infraestrutura oferece recursos para que ferramentas sejam executadas no servidor, como nos casos em que é necessário acesso direto a determinados recursos ou execução ininterrupta. Por exemplo, o "Desenvolvedor A" observa a necessidade de que o servidor execute uma tarefa diária para buscar dados online e alimentar a base de forma automática.

O objetivo dos requisitos apresentados não é determinar fatores obrigatórios para os componentes de *groupware*, mas uma base para facilitar a concepção dos componentes. Os requisitos apresentados por Tietze (2001) são definidos de forma generalizada para componentes de *groupware*. São muitos os possíveis tipos de *groupware*. Por isto, a seguir será apresentada a sistematização dos *groupwares* divididos em doze categorias funcionais definidas por Coleman (1997 apud ZOTTO, 2000):

1. Sistemas de correio eletrônico e comunicação
2. Agenda e calendário para grupos
3. Sistemas de reuniões eletrônicas
4. Desktop vídeo e conferência em tempo real (síncrono)
5. Conferência em tempo não real (assíncrono)
6. Gerenciamento de documentos multiusuários
7. Workflow
8. Utilitários workgroup e ferramentas de desenvolvimento
9. Serviços groupware
10. Frameworks groupware
11. Aplicativos groupware
12. Produtos e aplicativos colaborativos baseados na Internet

A seção a seguir aborda o padrão de projeto *observer*, destacando suas características e sua estrutura para aplicação.

2.2 PADRÃO DE PROJETO OBSERVER

Muitas aplicações mantêm uma forte dependência entre objetos, o que torna necessário o gerenciamento de estados entre esses objetos. Assim, para atender esta demanda pode-se utilizar o padrão de projeto *observer*, que tem por objetivo "definir uma dependência um-para-muitos entre objetos, de maneira que quando um objeto muda de estado todos os seus dependentes são notificados e atualizados automaticamente" (GAMMA et al., 2000, p. 274).

Com o intuito de exemplificar, suponha o seguinte cenário: em uma ferramenta colaborativa para criação de relatórios o “Usuário A” compartilha o mesmo ambiente que o “Usuário B” e ambos estão trabalhando sob a concepção do objeto “Tabela A” conforme a Figura 7.

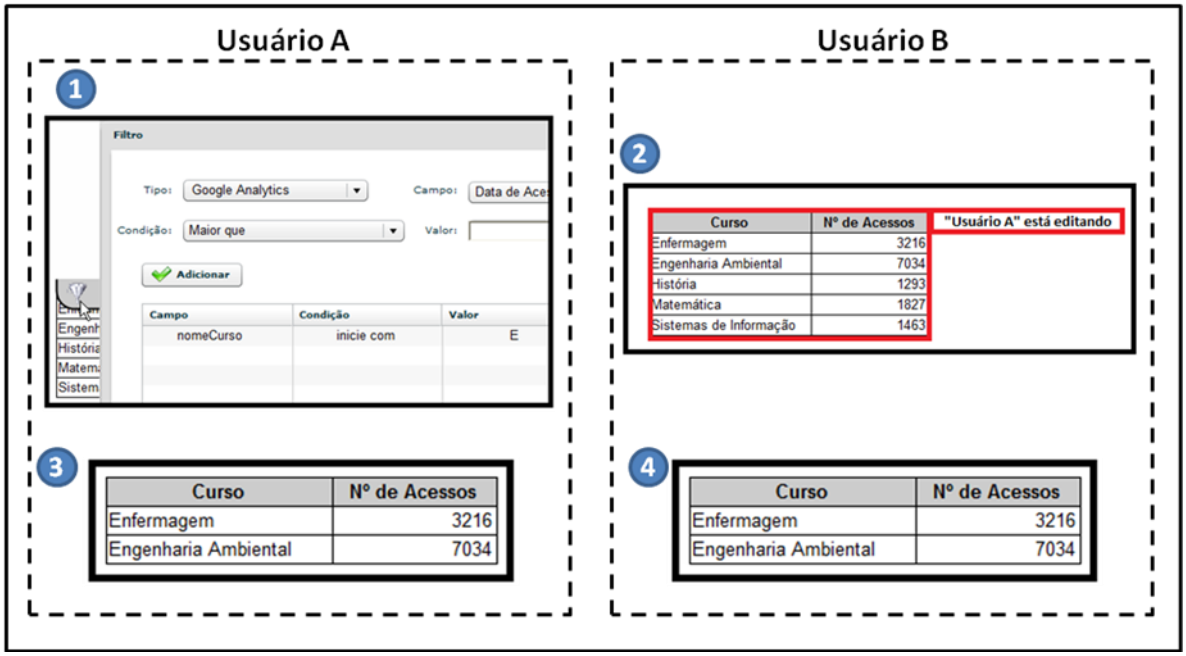


Figura 7 - Exemplo de Aplicação do Padrão de Projeto Observer

No cenário apresentado na Figura 7 pode-se perceber uma possível aplicação do padrão *observer*, pois cada usuário tem sua própria visualização da “Tabela A” e essa visualização é definida conforme as interações dos usuários sob a interface gráfica, se algo é alterado no ambiente do “Usuário A” isso deve ser refletido no ambiente do “Usuário B”, existindo assim uma dependência entre os objetos, conforme prevê a definição do padrão *observer*. No momento que o “Usuário A” muda o estado da “Tabela A” para edição e insere um novo filtro (Item 1 da Figura 7), a interface gráfica do “Usuário B” é informada, o usuário é então impedido de editar a tabela. Assim que o “Usuário A” termina a inserção do novo filtro, tanto a interface gráfica do “Usuário A” quanto a do “Usuário B” são informadas, desta forma ambos são atualizados para apresentar os mesmo valores (Itens 3 e 4 da Figura 7). Para que esse tipo de controle seja realizado o padrão *observer* é composto da seguinte estrutura.

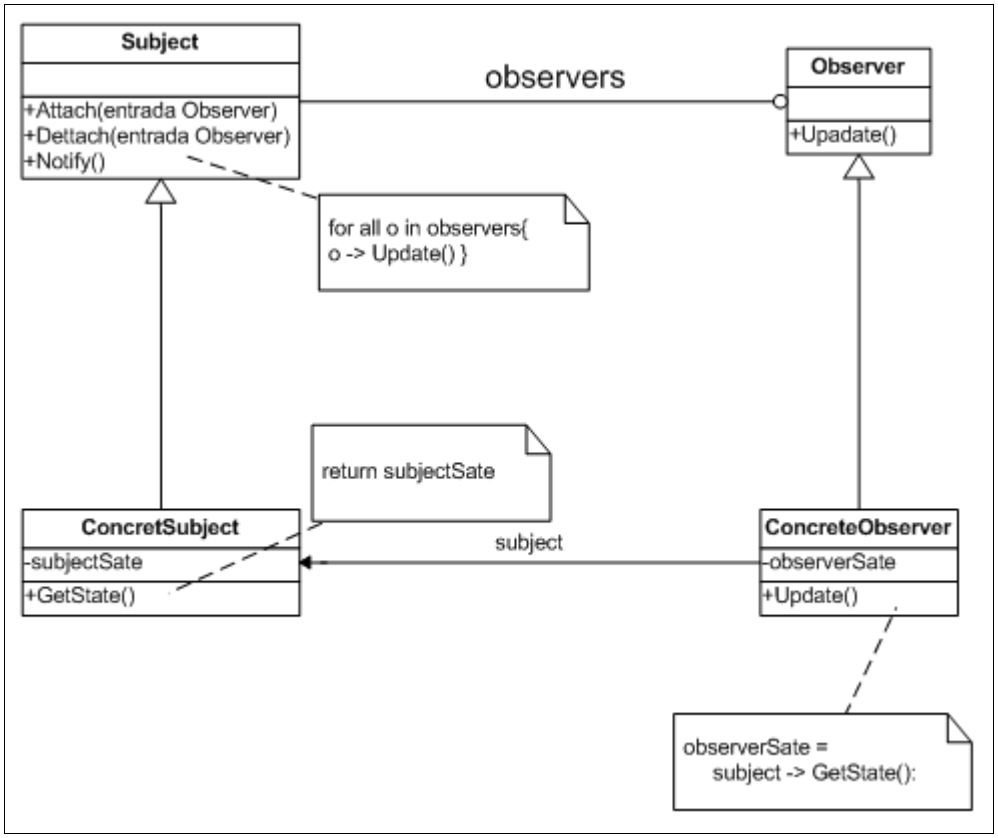


Figura 8 - Estrutura do Padrão de Projeto *Observer* (GAMMA et al., 2000, p. 275)

Analisando a estrutura do padrão de projeto *observer*, apresentado na Figura 8 é possível realizar a separação dos objetos desta forma:

- ***Subject*** – é uma interface que tem por objetivo manter uma lista de objetos observadores, fornecendo funções para adicionar (*Attach*(*Observer*)) e remover (*Dettach*(*Observer*)) esses objetos. Além disso, dispõe de uma função para notificação (*Notify*()), que é executada sempre que um novo estado é gerado, permitindo assim notificar os observadores sobre a mudança de estado;
- ***Observer*** – define uma interface contendo uma função para atualização (*Update*()) do observador, sempre que o *subject* mudar seu estado será indicada uma nova notificação. O observador não se preocupa em ficar checando a todo o momento os novos estados gerados pela aplicação, pois essa responsabilidade é centrada no *subject*;
- ***ConcreteSubject*** – representa a implementação da interface *Subject*, tornando-se responsável por armazenar os estados de interesse dos objetos *ConcreteObserver* e enviar notificação para os observadores quando seu estado é modificado. Desta maneira pode ser entendido como o "assunto" que controla as alterações de um domínio específico, por exemplo, pode haver um *ConcreteSubject* "Relatorio" e/ou "PlanoDeContas", sendo que

cada implementação destes *subjects* gerencia mudanças de estados específicas de cada domínio;

- ***ConcreteObserver*** – armazena estados que devem se manter consistentes com os estados do *Subject*, implementando a interface do *Observer* para criar sua função de atualização, de forma a garantir a consistência entre seus estados com os do *Subject*.

Para aplicação desta estrutura, utilizou-se para exemplificar um cenário de ferramenta colaborativa para criação de relatórios, conforme apresentado na Figura 9.

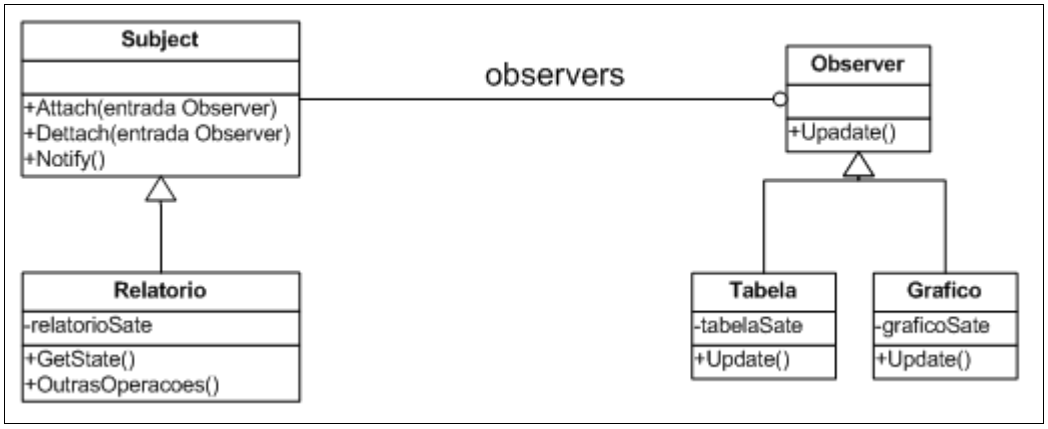


Figura 9 - Possível Estrutura para o Cenário Proposto

Na Figura 9 a classe *Relatorio* representa a implementação da interface *Subject*, sendo responsável por armazenar a lista de *observers* e notificá-los sempre que ocorrer uma atualização em alguma estação de trabalho. Já as classes *Grafico* e *Tabela* representam a implementação da interface *Observer*. A utilização do padrão *observer* para esse cenário se torna muito interessante, considerando que a qualquer momento novas classes podem ser acopladas ao projeto, como *TextoPlano* ou *Cabecalho*, sem que gere esforço para manter o trabalho da classe *Relatorio* de gerenciar as notificações para as novas classes, pois a estrutura do padrão *observer* tem baixo acoplamento, logo, a classe *Relatorio* lida diretamente com a interface *Observer* e independe das estruturas das novas classes. Para facilitar o entendimento dos passos que compõe o funcionamento do padrão *observer* para o cenário apresentado, tem-se o diagrama de sequência a seguir.

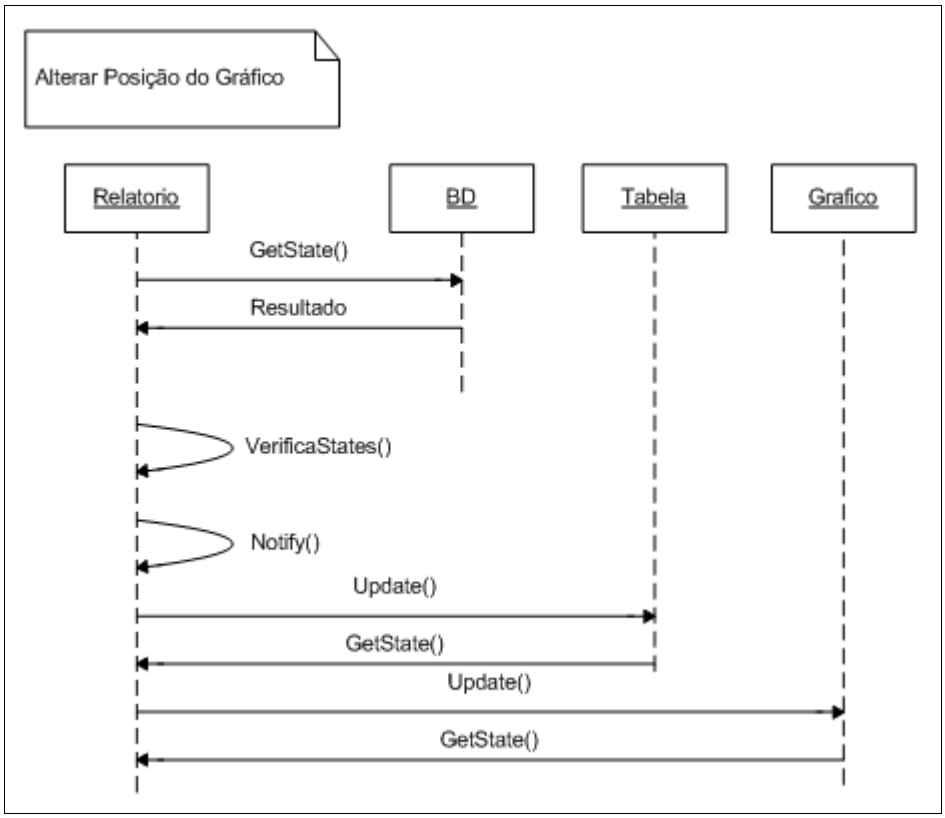


Figura 10 - Diagrama de Sequência do Cenário Proposto

Para descrever os passos apresentados na Figura 10, será simulada uma situação em que o "Usuário A" realiza a atualização de posição de um gráfico em seu ambiente cooperativo, assim, informações sobre essa operação são enviadas à base de dados da ferramenta. O "Usuário B" precisa ter sua tela atualizada para se manter sincronizado com o "Usuário A" e não atrapalhar o trabalho colaborativo. Para isso, a interface gráfica do "Usuário B" tem sua classe *Relatorio* que recupera seu estado através de consulta a base de dados (Passo 1 da Figura 10), funcionando como um serviço constante. Desta forma, a classe *Relatorio* do "Usuário B" irá verificar se houve alteração em seu estado (Passo 2), caso haja alteração, a classe irá percorrer sua lista de *observers* através da função *Notify*() (Passo 3), com propósito de notificá-los sobre o novo posicionamento do gráfico, para que os objetos *Tabela* e *Grafico* possam atualizar seu estado quando necessário (Passo 4 e 5 respectivamente). Vale ressaltar que a forma de recuperar o estado e disparar a notificação pode variar. Segundo Gamma et al. (2000), esse controle pode ser realizado pelo próprio *subject* ou pelo cliente.

Para Silva et al. (2009), o padrão *observer* apresenta as seguintes vantagens:

- **Permite adicionar observadores sem modificar a classe *Subject*** – Isso ocorre porque a classe *Subject* mantém uma lista de objetos *Observer* e não *ConcreteObserver*, o que torna possível tratar de forma genérica sem se preocupar com as implementações de cada objeto *Observer* (isto é, seus *ConcreteObservers*);
- **Baixo acoplamento entre a classe *Subject* e a interface *Observer*** – Tirelo et al. (2004, p. 7) destaca que o padrão *observer* é "utilizado para desacoplar o objeto alvo de observação de seus observadores, permitindo maior grau de reuso". Assim, as regras são definidas por cada observador, deixando o *Subject* responsável apenas por manter a lista de objetos *Observers* e notificá-los de modificações;
- **No momento que um objeto da classe *Observer* é notificado sobre uma atualização, ele tem a autonomia de considerá-la ou ignorá-la** – Esse é um fato que colabora para reduzir trabalho desnecessário, já que cada objeto *ConcreteObserver* implementa sua própria versão, ele pode considerar se a alteração é pertinente ou não para seu fim.

Segundo Gamma et al. (2000), algumas considerações sobre implementação devem ser realizadas:

1. **Mapeando *subjects* para os seus observadores** – A forma mais simples desse mapeamento é armazenar uma referência explicita dos observadores nos *subjects*, porém quando existem muitos *subjects* e poucos observadores pode ser muito dispendioso, nesses casos uma saída é manter mecanismos associativos como tabelas de acesso randômico, anulando o custo de memória para *subjects* sem observadores, porém aumenta o custo de acesso aos observadores;
2. **Observando mais que um *subject*** – Podem ocorrer casos em que um observador dependa de mais de um *subject*, por exemplo, um cenário que uma planilha com dados monetários, tenha dependência de seus dados em diversas fontes, como um site financeiro e uma base de dados local. Neste caso pode ser interessante o observador saber qual *subject* está enviando a notificação. Para isso, pode-se simplesmente estender a função *Update()* da interface *Observer* e acrescentar o próprio objeto do *subject* aos parâmetros;
3. **Quem dispara a atualização?** – Existem duas opções, na primeira a responsabilidade de controlar os estados é do *subject*, e tem por vantagem o fato de que os clientes não necessitam se lembrar de chamar a função de notificação no *subject*, a desvantagem é que várias trocas de estados consecutivas irão gerar diversas atualizações consecutivas, o que pode ser ineficiente. Na segunda opção a responsabilidade é mantida pelo cliente, tem por vantagem a possibilidade de juntar maior número de

estados para disparar a notificação, tornando mais eficiente, porém essa responsabilidade adicional do cliente pode gerar erros, uma vez que os clientes podem se esquecer de disparar a notificação;

4. **Referências ao vazio (*dangling references*) para *subjects* deletados** – A remoção de um *subject* não deve gerar esse tipo de referência em seus observadores, para isso uma opção é o *subject* notificar seus observadores sobre sua deleção. Não é uma opção deletar os observadores, pois estes podem estar relacionados com outros objetos e até mesmo estarem observando outros *subjects*;
5. **Garantir que o estado do *observer* é autoconsistente antes da emissão da notificação** – Essa garantia deve ser realizada, pois os observadores consultam os estados do *subject* durando o processo de atualização e caso o estado do *subject* modifique após a execução do método *Notify()* o observador ficará com seu estado inconsistente. Para resolver este problema e garantir a autoconsistência do *subject* pode-se utilizar o padrão de projeto *Template Method* que tem como objetivo "definir o esqueleto de um algoritmo em uma operação, postergando (*deferring*) alguns passos para subclasses [...] permite que subclasses redefinam certos passos de um algoritmo sem mudar a estrutura do mesmo" (GAMMA et al., 2000, p. 301). Assim, pode-se definir uma operação primitiva para ser substituída pelas subclasses e tornar a função de notificação a última operação a ser executada;
6. **Evitando protocolos de atualização específicos dos observadores: os modelos *push* e *pull*** – A implementação do padrão *observer* frequentemente tem o *subject* emitindo informações adicionais como parâmetro do método de *Update()*. Esse envio de informação pode ser dividido em dois extremos, no primeiro denominado *pull model*, o *subject* manda informações mínimas e posteriormente os observadores podem solicitar mais detalhes, enfatizando a ignorância dos *subjects* sobre seus observadores, podendo ser ineficiente, pois os observadores devem verificar o que mudou sem ajuda do *subject*. Já no segundo extremo, denominado *push model*, o *subject* manda informações de forma detalhada, assumindo assim que os *subjects* tem conhecimento das necessidades de seus observadores, porém isso pode tornar os observadores menos reutilizáveis, já que os *subjects* fazem hipóteses sobre seus observadores;
7. **Especificando explicitamente as modificações de interesse** – Para melhorar a eficiência do processo de atualização, pode-se estender o método *Attach()* da interface *Subject* permitindo que os observadores se registrem apenas em eventos específicos de

seu interesse. Quando o tal evento ocorrer, o *subject* informa apenas os observadores com interesse nesse tipo de evento;

8. **Encapsulando a semântica de atualizações complexas** – Em casos de dependências complexas entre *subjects* e observadores, pode ser necessário um objeto que mantenha esses relacionamentos. Para isso pode-se utilizar um objeto *ChangeManager* (Gerenciador de Mudanças), que fica responsável por manter o mapeamento entre *subjects* e observadores, definir estratégia de atualização e atualizar todos observadores a partir da solicitação de um *subject*. O objeto *ChangeManager* é uma instância do padrão Mediador que segundo Gamma et al. (2000, p. 257) esse padrão tem por objetivo "definir um objeto que encapsula a forma como um conjunto de objetos interage";
9. **Combinando as classes Subject e Observer** – Algumas linguagens como *Smalltalk*, que não apresentam herança múltipla, geralmente não dispõem das classes *Subject* e *Observer* de forma separada, tendo que combinar as interfaces em uma única classe.

A próxima seção aborda sobre a classe de banco de dados denominada NOSQL, apresentando suas características, principais categorias e seu impacto na resolução de problemas de escalabilidade.

2.3 NOT ONLY SQL (NOSQL)

Existem diversos SGBDs (Sistemas de Gerenciamento de Banco de Dados) disponíveis no mercado. Porém, tanto na literatura quanto em soluções comerciais, o modelo relacional de banco de dados é o mais difundido. Isto porque, segundo Lóscio et al. (2011, p. 3), "os conceitos básicos do modelo relacional são: relação (tabela), atributo (coluna) e tupla (linha)", os quais são fáceis de compreender. Por exemplo, em um sistema colaborativo para criação de relatórios, existe uma relação entre as tabelas "Grafico" e "Relatorio" (em que, um gráfico pertence a um relatório e um relatório pode possuir vários gráficos), a tabela "Grafico" é composta por atributos que formam seu conceito (como, posição, largura, altura, legenda) e por fim, as tuplas representam a instância desses atributos, assim cada atributo da tabela "Grafico" assume valores (por exemplo, "(34,21)", "150px", "275px", "Exemplo de Legenda").

Os bancos de dados relacionais surgiram no início dos anos 70, "os quais se firmaram como solução comercial para armazenamento e gerenciamento de dados convencionais, ou seja, dados que possuem uma estrutura fixa, bem definida e com tipos de dados simples"

(LÓSCIO et al., 2011, p. 3). Porém, com advento da Web 2.0 as necessidades em armazenamento de dados modificaram, pois são grandes volumes de dados produzidos continuamente, principalmente através de serviços de busca como Google Search, Bing, entre outros e redes sociais como Twitter, Facebook, MySpace etc. Em geral, estes dados não possuem estrutura fixa, como as páginas *web* (em que as estruturas descritas nos documentos HTML não indicam muito da semântica do conteúdo do documento) e materiais multimídia (como textos, vídeos, áudios etc.). Para suprir essa necessidade surgiu uma nova categoria de banco de dados denominada NOSQL (abreviação para "*Not Only* SQL"), que vieram como "uma solução para a questão da escalabilidade no armazenamento e processamento de grandes volumes de dados na *Web* 2.0" (DIANA & GEROSA, 2010, p. 2).

Os serviços oferecidos na *Web* 2.0 não podem parar, e empresas como Google trabalham com dados em uma escala de *pentabytes* e, independentemente desse crescimento maciço dos dados, é necessário manter a performance e a disponibilidade dos serviços para os usuários. Esse "gerenciamento de grande volume de dados tem se mostrado problemático, a utilização de SGBDs relacionais, ou em outras palavras do próprio Modelo Relacional, já não se mostra tão eficiente" (BRITO, 2010, p. 2). Assim, o NOSQL foi proposto "com o objetivo de atender aos requisitos de gerenciamento de grandes volumes de dados, semi-estruturados ou não estruturados, que necessitam de alta disponibilidade e escalabilidade" (LÓSCIO et al., 2011, p. 1). A Figura 11 apresenta uma relação do volume de dados com a complexidade das bases de dados relacionais e NOSQL.

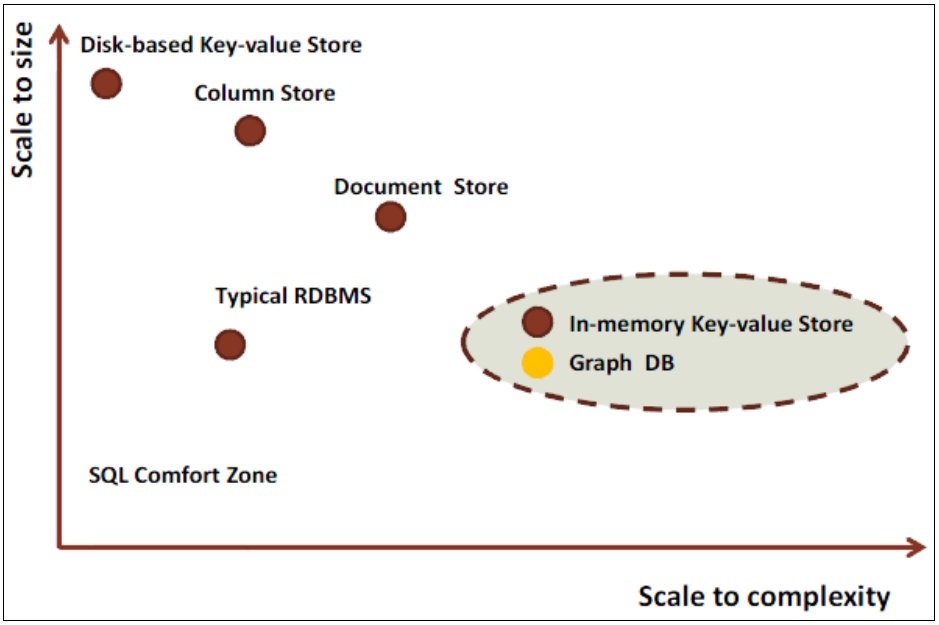


Figura 11 – Tamanho dos dados (EIFREM, 2009 apud SHAO et al., 2012, p. 2, imagem adaptada)

Pode-se observar, na Figura 11, a relação entre o tamanho e a complexidade dos dados, em que todos os itens (exceto os modelos relacionais típicos) tratam-se de bases de dados NOSQL. A escolha pelo modelo de banco de dados que melhor atenda a determinada solução é algo relativo. Por exemplo, os modelos relacionais (que compõe a zona de conforto da Figura 11) "amadureceram após mais de quarenta anos de melhorias contínuas, somente manipulam dados de tamanhos relativamente pequenos e de baixa complexidade" (SHAO et al., 2012, p. 2, tradução nossa), tornando-se um modelo muito estável, mas que não atende muitas das demandas da *Web* 2.0, que são centradas em grandes volumes de dados e exigência de flexibilidade quanto a estrutura dos dados. Ainda analisando a Figura 11 pode-se notar que as bases de dados NOSQL apresentam opções da relação do tamanho com a complexidade dos dados bastante variada, oferecendo uma gama maior de soluções para análise conforme a necessidade da aplicação.

Segundo Diana & Gerosa (2010), o termo NOSQL por si não contribui para o entendimento de seu real sentido, pois o termo SQL não é sinônimo do modelo relacional, nem representa limitações desses bancos de dados. Acima da nomenclatura deve-se analisar uma lista de características desejáveis desse modelo de banco de dados. Lóscio et al. (2011) apresenta uma lista com as principais características que tornam a classe de banco de dados NOSQL uma escolha adequada para o armazenamento de grandes volumes de dados não estruturados ou semi-estruturados:

- **Escalabilidade horizontal** – Com aumento do volume de dados, as aplicações podem enfrentar problemas com a falta de escalabilidade, que representa uma relação da quantidade de dados com a queda de performance da aplicação. Para solucionar esse tipo de problema pode-se trabalhar com a escalabilidade horizontal ou vertical. Na escalabilidade vertical a solução consiste em aumentar o poder de processamento e armazenamento das máquinas (montando servidores com mais memória RAM, HD e processadores mais potentes). Já a solução da escalabilidade horizontal consiste em aumentar o número de servidores processando e armazenando os dados. Porém para que se tenha a escalabilidade horizontal é necessário que diversos processos de uma tarefa sejam criados e distribuídos, o que é uma grande dificuldade para o modelo relacional, que tendo uma estrutura fixa precisa relacionar suas tabelas em instruções como *joins* (que realizam relações cartesianas entre as tabelas da base de dados). Uma solução conhecida é o *sharding*, em que "a ideia básica é distribuir o banco em várias máquinas, utilizando-se do particionamento dos dados" (BRITO, 2010, p. 4),

entretanto é uma solução muito complexa para bases relacionais. O NOSQL por não trabalhar com bloqueios apresenta soluções para escalabilidade horizontal de forma natural;

- **Ausência de esquema ou esquema flexível –** "A constatação de que os dados na web não são estruturados é um dos fatores que favoreceram o surgimento de tecnologias de gerenciamento de dados diferentes das tradicionais" (DIANA & GEROSA, 2010, p. 3). Atende a demanda da Web 2.0 de dados não estruturados ou semi-estruturados, além de facilitar aplicação da escalabilidade e disponibilidade dos dados. Porém não garante a integridade dos dados, o que é uma garantia encontrada em modelos relacionais (por possuírem estrutura rígida).;
- **Suporte nativo a replicação –** O suporte nativo a replicação contribui para escalabilidade, a replicação diminui o tempo gasto com a recuperação da informação. Pode-se dividir a replicação das informações em duas linhas:
    - **Master-Slave –** A escrita é realizada no nó mestre, que replica os dados para "N" nós escravos. Essa abordagem torna a leitura mais rápida, porém a capacidade de escrita pode ser comprometida para um grande volume de dados;
    - **Multi-Master –** Ao contrário do *Master-Slave*, não apresenta apenas um nó mestre, mas vários. Dessa forma situações de grande volumes de dados, que gera um gargalo na escrita no modelo *Master-Slave* é solucionado no *Multi-Master*, porém, a existência de diversos nós mestres pode gerar um problema de conflito de dados.
- **API simples para acesso aos dados –** O foco dos bancos NOSQL prioriza o fácil acesso aos dados, eficiência na recuperação dos dados, promovendo alta disponibilidade e escalabilidade. Para isso as APIs devem oferecer a facilidade de manuseio para que qualquer aplicação possa adotar a solução de bases NOSQL de forma rápida e eficiente;
- **Consistência eventual – "**O sistema de armazenamento garante que se nenhuma nova atualização for realizada sobre o objeto, eventualmente todos os acessos a esse objeto retornarão o último valor atualizado" (BRITO, 2010, p. 5). Logo, o sistema deve permitir inconsistências temporárias a fim de priorizar a disponibilidade dos dados. Nesse sentido, a consistência só é garantida se outra operação não estiver sendo executada ao mesmo tempo.

Tabela 1 – **Análise Comparativa Modelo Relacional x NOSQL (BRITO, 2010, p. 5)**

	Relacional	**NOSQL**
Escalonamento	Possível, mas complexo. Devido à natureza estruturada do modelo, a adição de forma dinâmica e transparente de novos nós no *grid* não é realizada de modo natural.	Uma das principais vantagens desse modelo. Por não possuir nenhum tipo de esquema pré-definido, o modelo possui maior flexibilidade o que favorece a inclusão transparente de outros elementos.
Consistência	Ponto mais forte do modelo relacional. As regras de consistência presentes propiciam um maior grau de rigor quanto à consistência das informações.	Realizada de modo eventual no modelo: só garante que, se nenhuma atualização for realizada sobre o item de dados, todos os acessos a esse item devolverão o último valor atualizado.
Disponibilidade	Dada a dificuldade de se conseguir trabalhar de forma eficiente com a distribuição dos dados, esse modelo pode não suportar a demanda muito grande de informações do banco.	Outro fator fundamental do sucesso desse modelo. O alto grau de distribuição dos dados propicia que um maior número de solicitações aos dados seja atendida por parte do sistema e que o sistema fique menos tempo não disponível.

A Tabela 1 permite uma análise comparativa entre as classes de banco de dados, a partir do teorema CAP (*Consistency, Availability e Partition Tolerance)*, que em português, significa: consistência, disponibilidade e tolerância ao particionamento (na Tabela 1 representada pelo escalonamento). Essas três características são esperadas em um sistema computacional distribuído, porém segundo o teorema, no mundo real, um sistema computacional distribuído só pode garantir apenas duas dessas características simultaneamente (Leite, 2010). Essa abordagem pode ser observada em bases NOSQL, pois, apesar das bases de dados NOSQL não oferecerem a propriedade de consistência como em bases relacionais (que garantem a integridade dos dados), apresenta como pontos fortes o escalonamento e a disponibilidade (conforme visto na Tabela 1). Pode-se concluir que a classe de bando de dados NOSQL não tem por objetivo substituir o modelo relacional, mas atender uma nova demanda do mercado, em situações em que seja necessário de forma flexível trabalhar com cargas de dados semi-estruturadas ou não estruturadas e também em sistemas que apresentem grandes volumes de dados e necessitem de disponibilidade para seus usuários.

Segundo Diana & Gerosa (2010), são “os tipos mais comuns de bancos de dados NOSQL: bancos de dados orientados a documentos, armazéns de chave-valor, bancos de dados de famílias de colunas e bancos de dados de grafos”. As seções a seguir detalham cada um desses tipos de banco de dados NOSQL.

2.3.1 Chave-valor

Este modelo tem sua composição de maneira simples, “trata-se de uma abordagem parecida com uma tabela *hash*” (TOTH, 2011, p. 3). Segundo Lóscio et al. (2010, p. 6), este “banco de dados é composto por um conjunto de chaves, as quais estão associadas um único valor, que pode ser uma *string* ou um binário”. A Figura 12 apresenta um exemplo de aplicação do modelo chave-valor.

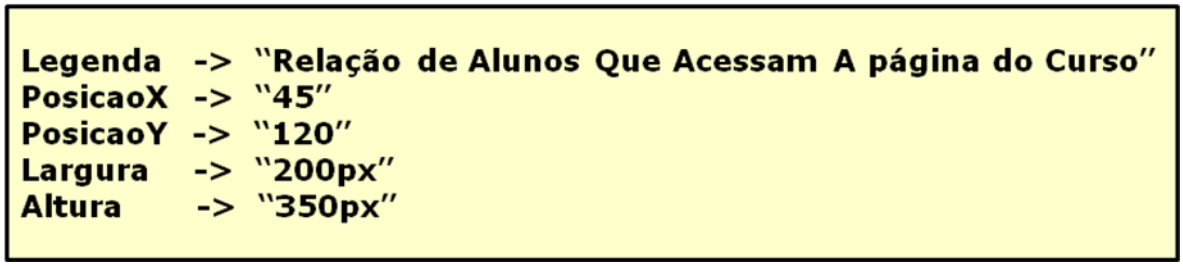


Figura 12 - Exemplo do modelo chave-valor

O exemplo apresentado na Figura 12 representa a estrutura de um armazém chave-valor e mantém os dados relativos a gráficos, em que as chaves são representadas pelos campos “Legenda” e “PosicaoX”, por exemplo, e os valores pela instância de seus respectivos campos (a chave “PosicaoX” tem valor “45” conforme Figura 12). O modelo chave-valor armazena estruturas simples, por não trabalhar com relações entre os dados, pode ocorrer muita replicação desnecessária em sua aplicação, por exemplo, no modelo apresentado na Figura 12 se fosse acrescentado a chave “Curso” seu valor deveria ser replicado para todos os gráficos pertencentes a esse curso, ao contrário, por exemplo, do modelo relacional que oferece recursos para relacionar um mesmo curso para diversos gráficos.

Segundo Strauch (2012, p. 52, tradução nossa), o modelo chave-valor “favorece alta escalabilidade ao invés da consistência e, portanto, a maioria deles também omite recursos ricos para consultas e ferramentas analíticas (especialmente operações de *joins* e agregação são postas de lado)”. Segundo Toth (2011), o modelo chave-valor na maioria dos casos apresenta uma interface composta principalmente pelos seguintes métodos:

- Put(chave, valor) – representa o método que insere um registro, garante maior velocidade a operação;
- Get(chave) – responsável por recuperar um registro, não oferece opções para buscas mais complexas.

A seção a seguir aborda sobre o modelo NOSQL orientado a colunas e suas principais características.

2.3.2 Orientado a Colunas

Este modelo se contrapõe ao paradigma do modelo relacional de armazenamento em tuplas (instâncias dos atributos em formato de linha), pois no modelo baseado em família de colunas "os dados são indexados por uma tripla (linha, coluna e *timestamp*), onde linhas e colunas são identificadas por chaves e o *timestamp* permite diferenciar múltiplas versões de um mesmo dado" (LÓSCIO et al., 2011, p. 6). A Figura 13 apresenta um exemplo para distinção do modelo tradicional em linhas com o modelo baseado em colunas.

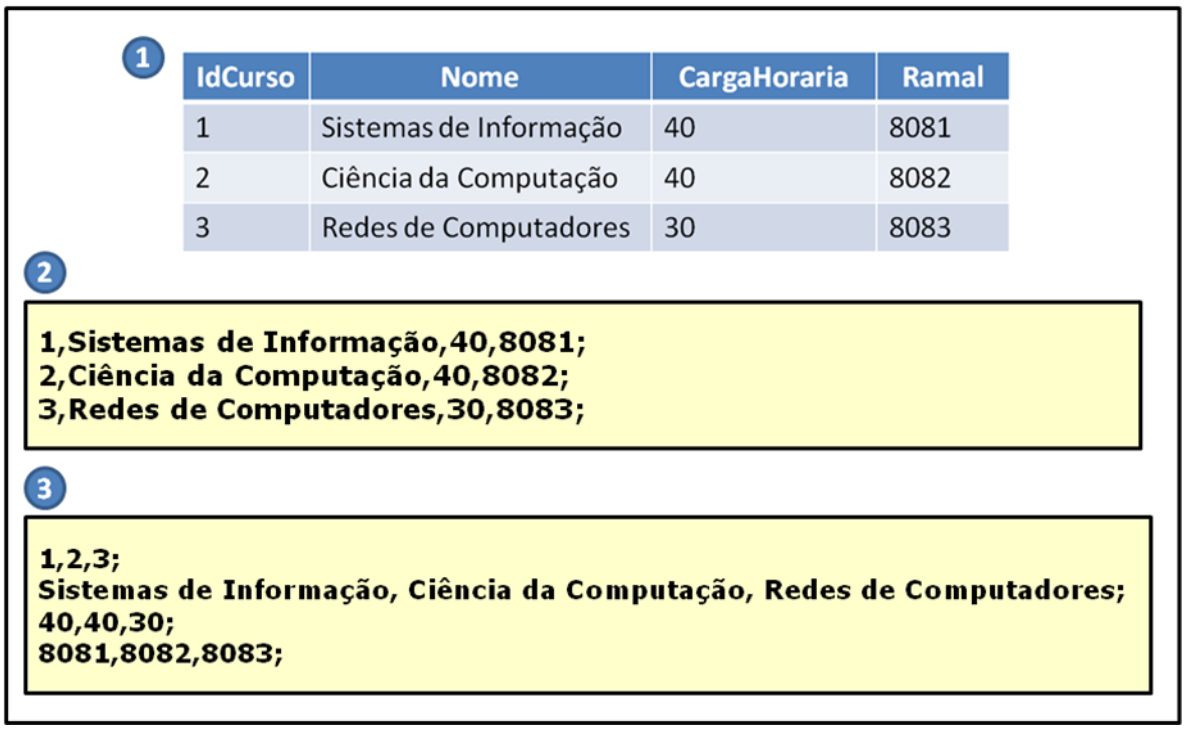


Figura 13 - Diferença do armazenamento orientado a linha e a coluna

O Item 1 (Figura 13) representa um modelo lógico de dados referentes a cursos de uma instituição de ensino. Pode-se observar no Item 2 como esses dados são alocados por linha seguindo o modelo relacional, em contra partida, no Item 3 os mesmos dados são

armazenados por colunas. Para melhor entendimento do modelo baseado em colunas, Silva (2012) utiliza um banco de dados denominado “Cassandra” e separa a estrutura da seguinte forma:

- Colunas: compostas por um nome que identifica a coluna, além de um valor e um *timestamp* (que são fornecidos pela aplicação cliente no momento da inserção).
- Família de colunas: é análogo a uma tabela do modelo relacional, ao contrário das colunas as famílias de colunas não são dinâmicas, sendo necessário declará-las anteriormente em um arquivo de configuração.
- Super Colunas: são compostas por outras colunas.
- Super Famílias de Colunas: são compostas somente por Super Colunas.

Segundo Abadi (2008), se o objetivo do trabalho é acessar dados sobre a granularidade de uma entidade (por exemplo, pesquisar um curso, inserir um novo curso, excluir um curso) então o modelo tradicional baseado em linhas é o mais indicado, considerando que todas as informações desejáveis são armazenadas em conjunto. Já em situações em que se necessita ler apenas alguns atributos de vários registros então o modelo de colunas é preferível (por exemplo, uma consulta que verifica a faixa etária mais comum no curso de Sistemas de Informação).

Diana & Gerosa destacam que o custo de escrita do modelo baseado em família de colunas é bem maior que em um banco de dados relacional, em que “os bancos tradicionais são mais adequados a processamento de transações online (OLTP) enquanto os bancos de dados de famílias de colunas são mais interessantes para processamento analítico online (OLAP)” (DIANA & GEROSA, 2010, p. 6).

A seção a seguir aborda sobre o modelo NOSQL orientado a grafos e suas principais características.

2.3.3 Orientado a Grafos

Segundo Diana & Gerosa (2010, p. 6), “diferentemente de outros tipos de bancos de dados NOSQL, esse está diretamente relacionado a um modelo de dados estabelecido, o modelo de grafos”. Esse modelo é muito difundido em outras áreas da computação e está relacionado a soluções matemáticas. A estrutura do modelo de grafos é composta por: “nós (são os vértices do grafo), os relacionamentos (são as arestas) e as propriedades (ou atributos) dos nós e relacionamentos” (LÓSCIO et al., 2011, p. 8). Toth (2012, p.4) ainda destaca que “pode-se

armazenar qualquer tipo de dados dentro do conteúdo". A Figura 14 apresenta um exemplo de aplicação do modelo orientado a grafos.

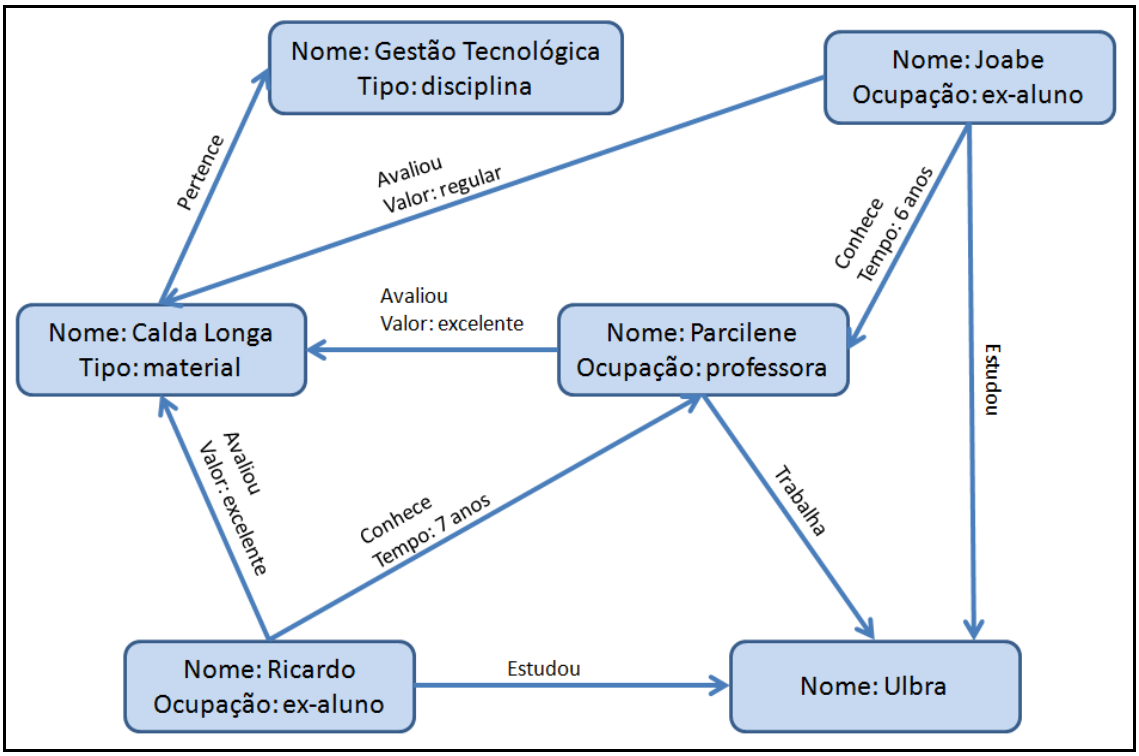


Figura 14 – Exemplo do modelo de grafo

É possível observar na Figura 14 que, ao contrário do modelo relacional que poderia gerar consultas muito mais complexas devido as operações de *joins* (que podem levar à diminuição de performance da aplicação), nos grafos essas consultas tornam-se mais simples e diretas. Por exemplo, uma possível consulta para ser aplicada no modelo de grafo apresentada na Figura 14 seria: "Quais pessoas que já estudaram na Ulbra consideraram regular um material da disciplina de Gestão Tecnológica?". Para esta situação uma das possíveis formas de se percorrer os nós é verificar apenas os materiais vinculados à disciplina de "Gestão Tecnológica". Assim, após recuperar o nó "Calda Longa", são verificados todos que avaliaram o material como regular, operação essa que terá como retorno o nó "Joabe". Por fim, será checado se o nó "Joabe" estudou na "Ulbra". "Não há um padrão para sistemas baseados em grafos, em que os algoritmos dos grafos são desenvolvidos e otimizados" (SHAO et al., 2012, p. 1). Porém, são vários os algoritmos existentes para percorrer um grafo de forma otimizada, por exemplo, algoritmos heurísticos conhecidos como: "*Best First*", "*Hill Climbing*" etc..

A seguir são listadas algumas das vantagens na utilização do modelo orientado a grafo:

- O fato de “não existir tanta replicação de dados como nos outros modelos, fato que acontece por se aproveitar do relacionamento entre os registros” (TOTH, 2012, p. 4). Em modelos como o orientado a documentos, por exemplo, cada documento armazena todas suas relações de forma atômica, desta forma, se houver dois documentos, um representando a pessoa “Ricardo” e outra representando a pessoa “Joabe” e ambos conhecem a pessoa “Parcilene”, tanto o documento “Ricardo” quanto “Joabe” teriam que guardar uma instância da pessoa “Parcilene”. No caso dos grafos, isso é resolvido apenas utilizando ligações com arestas que armazenam um valor que indica que a pessoa conhece a “Parcilene”;
- “A vantagem de utilização do modelo baseado em grafos fica bastante clara quando consultas complexas são exigidas pelo usuário. Comparado ao modelo relacional, que para estas situações pode ser muito custoso” (LÓSCIO et al., 2010, p. 8). Isso ocorre por existirem algoritmos heurísticos que podem ser aplicados para otimizar a busca por valores ao se percorrer um grafo, definindo o melhor caminho diretamente do nó inicial até o nó destino, sem que seja necessário verificar outros nós. No modelo relacional essas consultas exigiriam a construção de *joins*, que dependendo da complexidade podem afetar no desempenho de execução da consulta;
- “Esse modelo também dá suporte ao uso de restrições sobre os dados, como restrições de identidade e de integridade referencial, por exemplo” (Diana & Gerosa, 2010, p. 6). Ao contrário da maioria dos modelos NOSQL que são mais flexíveis;
- “Esse modelo pode tornar consultas rápidas, devido à possibilidade da utilização de propriedades de grafos, medidas de centralidade (*pagerank*) e algoritmos de menor caminho” (TOTH, 2012, p. 3).

Logo, para situações de contexto complexo em que se conhece a semântica entre os objetos armazenados, o modelo orientado a grafo pode apresentar-se como uma escolha vantajosa. A próxima seção aborda sobre o modelo NOSQL orientado a documentos e suas principais características.

2.3.4 Orientado a Documentos

Esse modelo de banco de dados armazena coleções de documentos, em que um documento representa “um objeto com um identificador único e um conjunto de campos, que podem ser *strings*, listas ou documentos aninhados” (LÓSCIO, 2011, p. 7). Diana & Gerosa (2010, p. 5) destacam que as bases de dados orientadas a documentos “não possuem esquema, ou seja, os

documentos armazenados não precisam possuir estrutura em comum", isto permite que novos campos sejam adicionados a um documento sem que isso cause algum problema na base de dados. A Figura 15 apresenta uma comparação entre dados armazenados em uma base de dados relacional e armazenados em uma base de dados orientada a documentos.

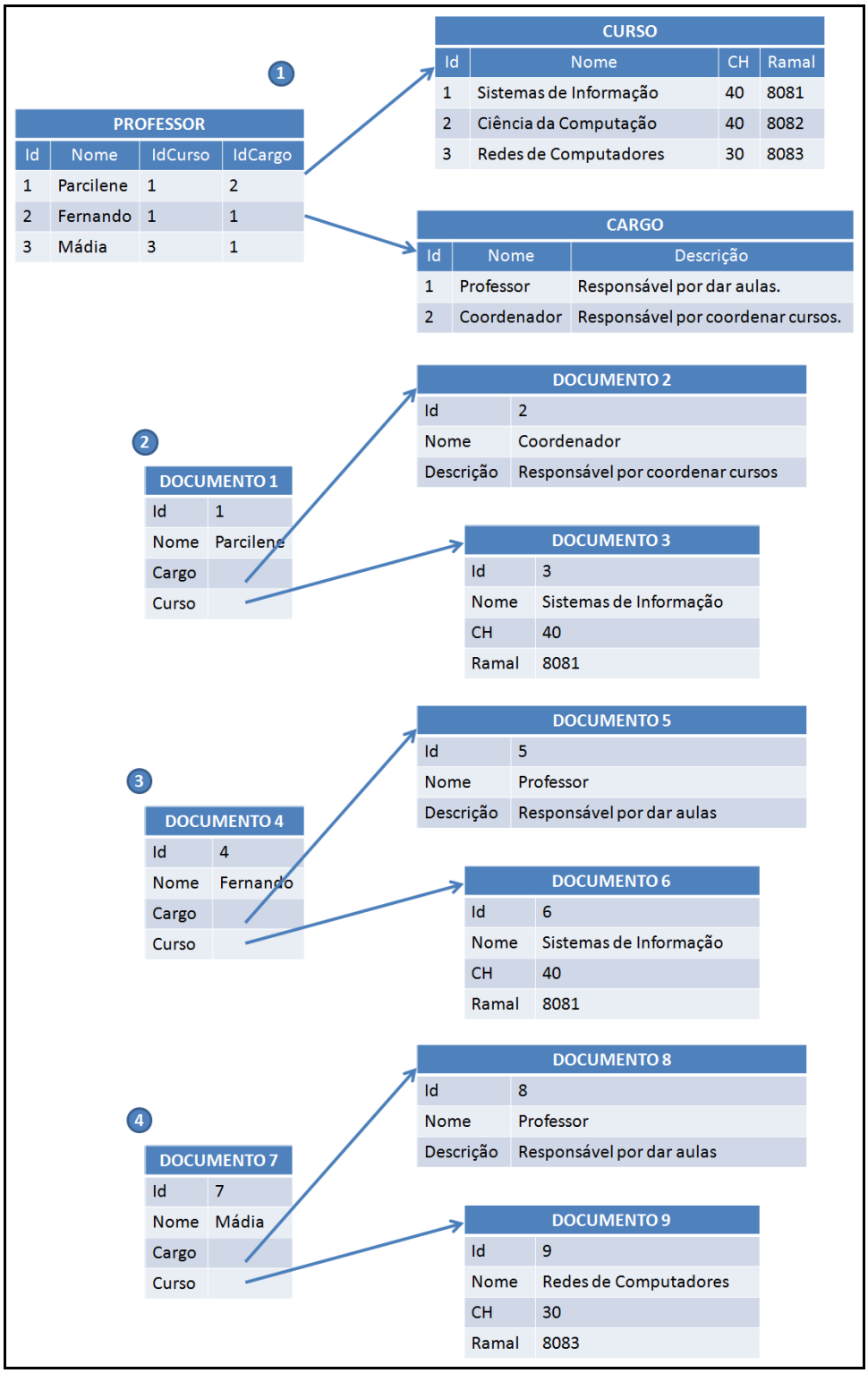


Figura 15 – Exemplo do modelo de documentos

É possível observar a partir da Figura 15 um cenário que controla dados de pessoas que trabalham em uma instituição de ensino, em que as pessoas possuem cargo e estão relacionadas a um curso. A seguir será apresentada uma explicação (baseada na Figura 15) da diferença de armazenamento entre o modelo relacional e o modelo orientado a documentos:

- **Modelo Relacional** – Representado pelo Item 1, existe uma tabela Pessoa, que relaciona uma Pessoa a um Curso e um Cargo. As consultas são baseadas em *joins* e a estrutura é rígida, não permitindo modificações sem que afete todas as instâncias cadastradas;
- **Modelo Orientado a Documentos** – Representado pelos Itens 2, 3 e 4. Cada um destes documentos representa uma pessoa, assim como a tabela "Pessoa" do modelo relacional (Item 1), porém as relações com curso e cargo estão relacionadas dentro do próprio documento que guarda os dados de "Pessoa". Pode-se perceber nos documentos das pessoas "Parcilene" e "Fernando" (Itens 2 e 3) que cada documento desses armazena um documento correspondente ao mesmo curso "Sistemas de Informação", porém, esses documentos de curso, apesar de apresentarem o mesmo valor, são independentes para "Parcilene" e "Fernando", isto é, caso o documento de curso da "Parcilene" mude a carga horária para "20", isso não afetará o documento curso do "Fernando", em que a carga horária permanecerá "40". Ao contrário do modelo relacional que tem sua estrutura rígida, no modelo orientado a documentos não há uma estrutura definida, o que permitiria, por exemplo, que no documento "Mádia" fosse acrescentado o campo email sem que afetasse os outros dois documentos (Itens 2 e 3).

O fato dos documentos integrarem suas relações diretamente dentro do próprio documento é o fator que gera duplicidade dos dados, porém torna a consulta direta. Desta forma, por exemplo, tendo como contexto a Figura 15, caso fosse criada uma consulta para recuperar o cargo da pessoa Parcilene, no modelo orientado a documentos bastaria recuperar o documento da Parcilene, que conteria o documento de seu cargo, já no modelo relacional seria necessário realizar um *join* entre a tabela Pessoa e Cargo, o que tem um custo de desempenho para execução, melhor percebido em consultas mais complexas. Outra vantagem da duplicação de dados é que "facilita a distribuição do sistema, já que a quantidade de nós a serem consultados em uma busca envolvendo várias entidades relacionadas é menor caso elas estejam próximas" (DIANA & GEROSA, 2010, p.5). Porém, dependendo do contexto, segundo Diana & Gerosa (2010, p. 5), "pode criar problemas de consistência no banco de dados, causados por anomalias de atualização e deleção", isso ocorre, por exemplo, caso um documento duplicado seja atualizado em apenas um documento que o armazena.

Segundo Strauch (2012), as bases de dados Apache CouchDB e MongoDB são os principais representantes dessa classe de banco de dados baseada em documento. A seção a seguir aborda o banco de dados MongoDB que foi utilizado no desenvolvimento do projeto proposto no presente trabalho.

2.3.4.1 MONGODB

Segundo Matteussi (2010, p. 39), "MongoDB é um banco de dados orientado a documentos de alta performance, *open source* e de esquema livre, escrito em C++. Formado por uma mistura entre os repositórios escaláveis e a tradicional riqueza de funcionalidades dos bancos de dados relacionais". Os documentos contemplados pela base de dados MongoDB é persistido "por um formato chamado BSON que é muito semelhante ao JSON mas em uma representação binária, por razões de eficiência e por causa de tipos de dados adicionais em comparação com JSON" (STRAUCH, 2012, p. 77). O modelo JSON é muito difundido no desenvolvimento *Web*, pois é bastante utilizado em soluções Ajax em aplicações que dependem de comunicações assíncronas. A Figura 16 apresenta como é estruturado um documento sob a notação JSON.

```
{
Nome : "Parcilene",
Cargo : { Nome: "Coordenador", Descricao: "Responsável por coordenar cursos"}
Curso : { Nome: "Sistemas de Informação", CH: "40", Ramal: "8081"}
}
```

Figura 16 - Exemplo de notação JSON

A Figura 16 apresenta um cenário em que os dados da pessoa "Parcilene" estão armazenado como um objeto no formato JSON, nesta situação, o objeto "Parcilene" contempla outros dois objetos Curso e Cargo. Através dos recursos oferecidos pela API do banco de dados MongoDB seria muito simples recuperar os dados da pessoa "Parcilene", a Figura 17 ilustra essa consulta.

```
<?php
$mongo = new MongoDB();
$db = $mongo->getDB();
$pessoa = $db->pessoa->find(array('nome' => "Parcilene"));
?>
```

Figura 17 - Exemplo de consulta no banco de dados MongoDB

Camara & Garcia (2011) sugere que os seguintes recursos do MongoDB sejam considerados:

- **Binários disponíveis para Linux, Sun Solaris, Apple MacOS e Microsoft Windows;**
- **Shell online, com possibilidade de testes sem a necessidade de instalação**;
- **Documentação abrangente e detalhada** – diminuindo assim a curva de aprendizagem;
- ***Drivers* oficiais para C, C++, C#, Haskell, Java, JavaScript, Perl, PHP, Python, Ryby e Scala, além de *drivers* para diversas outras linguagens suportados pela Comunidade –** O que demonstra a estabilidade e procura do banco de dados MongoDB como solução de armazenamento;
- **Suporte a expressões regulares em consultas** – Desta forma permite a realização de consultas mais complexas;
- **Escala horizontal com auto-sharding –** Permite que *clusters* sejam contados de forma dinâmica através de recursos oferecidos pela API do banco de dados MongoDB;
- **Opções de filtragem, agregação e classificação, tais como limit(), skip(), sort(), count(), distinct() e group()** – Recursos comuns no modelo relacional de banco de dados;
- **Armazenamento de grandes arquivos com uso de GridFS** - É "um mecanismo para armazenar grandes objetos eficientemente, visto que um objeto no formato BSON é limitado a 4MB e dividido em vários documentos" (SANTOS, 2011, p. 28);
- **Suporte a indexação com índices semelhantes aos do modelo relacional de banco de dados** – "a utilização de BSON habilita a indexação de qualquer tipo de dados, proporcionando o melhoramento do desempenho relativo às consultas" (SANTOS, 2011, p. 27);
- **Replicação Master/Slave** – "A leitura torna-se mais rápida, porém a capacidade de escrita torna-se um gargalo nesta abordagem" (LÓSCIO, 2011, p. 4).

Segundo os desenvolvedores do banco de dados MongoDB o seu principal objetivo é fechar a lacuna entre o rápido e altamente escalável modelo chave-valor e os bancos de dados relacionais que são ricos em funcionalidades tradicionais (STRAUCH, 2012, p.76). Deve-se destacar que esse banco de dados não substitui a utilização dos bancos de dados relacionais, muitas das aplicações necessitam, por exemplo, de uma estrutura rígida e/ou alta consistência. Porém, quando é preciso aplicar uma solução que utilize banco de dados orientado a documentos, o MongoDB pode ser uma boa opção, pois além de apresentar uma documentação abrangente e detalhada, também oferece recursos avançados de consulta (como

filtragem, agregação e classificação), o que torna a curva de aprendizagem menor para desenvolvedores que têm experiência na utilização do modelo relacional.

A seção a seguir apresenta as considerações sobre a revisão de literatura do presente trabalho.

2.4 CONSIDERAÇÕES

No decorrer da seção de revisão de literatura foram apresentados os conceitos sobre Sistemas Colaborativos, o Padrão de Projeto *Observer* e a classe de banco de dados NOSQL. Todos os conceitos foram abordados visando ao final aplicá-los como solução em um protótipo de sistema colaborativo que trata dados analíticos.

Considerando que um sistema colaborativo deve oferecer comunicação, coordenação e cooperação, o Modelo 3C de colaboração é eficiente, pois sua abordagem relaciona essas três dimensões e define a forma de interação entre elas. Deve-se ter uma atenção especial na definição dos *groupwares* de um sistema colaborativo, pois devem ser analisados sob aspectos humanos (cultura da equipe, estrutura da organização) e tecnológicos (por exemplo, suporte a comunicação, suporte a trabalho em ambiente de cooperação, baixa de performance etc.). Os requisitos definidos por Tietze (2001) apresentam pontos importantes para implementação de *groupware*, porém, não devem ser considerados como fatores obrigatórios, mas uma base para facilitar a concepção dos componentes.

O padrão de projeto *observer* pode ser utilizado em ambientes em que se necessite controlar as dependências entre os objetos. O modelo atende tanto aplicações complexas quanto simples, quando as dependências entre objetos se dá de forma complexa é possível unir o padrão *observer* a outros padrões (como *ChangeManager* e *TemplateMethod*, por exemplo). Uma das principais vantagens deste padrão de projeto é que o controle de notificação é centralizado na classe *Subject*, o que permite que posteriores adições de novos tipos de objetos se dê forma simplificada, já que o novo objeto estende da classe *Observer*, o que torna necessário apenas acrescentar o novo objeto como observador do *Subject* e implementando sua função *Update()* sem que isso afete os outros observadores que já existiam na aplicação.

A classe de banco de dados NOSQL pode ser considerada como uma boa solução para ambientes que apresentem problemas de escalabilidade, disponibilidade e flexibilidade para armazenar dados semi-estruturados ou até mesmo não estruturados. É importante destacar que a classe de bando de dados NOSQL não tem por objetivo substituir o modelo relacional, mas

atender uma nova demanda do mercado, em situações em que seja necessário trabalhar com cargas de dados semi-estruturadas ou não estruturadas e também em sistemas que apresentem grandes volumes de dados e necessitem de disponibilidade para seus usuários.

Na próxima seção serão apresentados os materiais e métodos utilizados no desenvolvimento deste trabalho.

3 MATERIAIS E MÉTODOS

Nesta seção serão apresentados os recursos e métodos utilizados durante o desenvolvimento deste trabalho.

3.1 Local e Período

O desenvolvimento deste trabalho deu-se nas dependências do complexo de informática do CEULP/ULBRA, bem como em residência própria, ambos localizados na cidade de Palmas-TO. O período de desenvolvimento deste trabalho ocorreu durante o primeiro semestre de 2012, como parte das disciplinas “Trabalho de Conclusão de Curso I” e “Trabalho de Conclusão de Curso II”.

3.2 Materiais

Para que fosse possível a realização deste trabalho foram utilizados recursos próprios e outros adquiridos através da internet. Dentre os materiais utilizados no referencial teórico estão: artigos, trabalhos de conclusão de curso, dissertações de mestrado e doutorado, livros que abordam o assunto e *sites*.

3.3 Métodos

Em um primeiro momento buscou-se um aprofundamento referente aos conceitos relacionados ao trabalho em questão. Para a obtenção do entendimento sobre o tema do referente trabalho foram realizadas pesquisas sobre Sistemas Colaborativos, Modelo 3C de Colaboração, *Groupware*, Padrão de Projeto *Observer* e NOSQL, sempre com objetivo de aplicar os aspectos conceituais nas soluções propostas para o protótipo apresentado neste trabalho.

Com o término dos estudos teóricos, foi dado início à produção dos artefatos necessários para o desenvolvimento do protótipo. Deve-se considerar que a primeira versão da ferramenta é encontrada em Almeida (2011) e muito dos artefatos foram aproveitados e/ou estendidos no presente trabalho. Inicialmente, o artefato “Casos de Uso Expandido” estava previsto. Entretanto, com o detalhamento descritivo do *storyboard*, foi decidido em comum acordo com a orientadora do trabalho, pela exclusão deste artefato.

Primeiramente foi definido o escopo do protótipo, deixando claro quais são os limites que o protótipo irá abordar, centrando as funcionalidades desenvolvidas na comprovação da utilização do Modelo 3C de Colaboração, Padrão de Projeto *Observer* e Classe de banco de dados NOSQL como soluções para ferramenta proposta.

Após a definição de escopo foi realizado o levantamento de requisitos, em que foram levadas em consideração as boas práticas indicadas nos requisitos de desenvolvimento e usuário para *groupware*.

Com os requisitos concluídos foram desenvolvidos três documentos justificando e delimitando a utilização do Modelo 3C, Padrão de Projeto *Observer* e NOSQL como soluções para o protótipo. Em seguida foi desenvolvida a arquitetura do sistema.

Passada a fase de definição da arquitetura do sistema, foi desenvolvido um *storyboard* do protótipo, com propósito de aumentar o entendimento da ferramenta e esboçar as características das telas do sistema. Após a finalização do *storyboard*, iniciou-se a análise de possíveis tecnologias para o desenvolvimento da ferramenta.

Considerando que a ferramenta teve sua versão anterior definida em Almeida (2011), as ferramentas e tecnologias necessárias para manipular recursos avançados como "*drag and drop*" já estavam instaladas e não houve alteração no ambiente de desenvolvimento. Dentre as tecnologias, a única inserida foi o banco de dados MongoDB, que foi a base escolhida para armazenamento das informações importadas de bases de dados relacionais pelo usuário.

Com a conclusão do *storyboard* foi possível visualizar todos os campos de cada funcionalidade, o que permitiu a construção do modelo de banco de dados relacional da ferramenta. Essa base relacional tem por objetivo armazenar os dados referentes à construção da estrutura dos relatórios, armazenamento dos estados de alterações, controle de usuários e gerenciamento dos projetos.

Por fim, as atenções foram voltadas para implementação da ferramenta. Inicialmente foi desenvolvido o método responsável por converter a base relacional para um banco de dados orientado a documentos. Após isso, aplicou-se o padrão de projeto *observer* no protótipo. Em seguida, criou-se a interface gráfica, com simulação de dados estáticos e, então, foram vinculados dados reais advindos dos *datasources* originados da importação das bases de dados relacionais para o banco de dados orientado a documentos.

4 RESULTADOS E DISCUSSÃO

Através da proposta de um protótipo de ferramenta colaborativa para criação de relatórios a partir de diversas estruturas de dados, foram definidos os artefatos necessários para o entendimento do projeto. As seções, a seguir, apresentam os artefatos produzidos durante o desenvolvimento do protótipo, assim como os resultados deste desenvolvimento.

4.1 ESCOPO

O objetivo do protótipo é permitir que, através de soluções colaborativas, o protótipo sirva de apoio à geração de relatórios, disponibilizando recursos para coordenação, comunicação e cooperação, que ofereça flexibilidade no que tange a possibilidade do usuário indicar diversas estruturas de dados para trabalhar. O presente trabalho dá continuidade a ferramenta concebida em Almeida (2011), cujo enfoque era exclusivo na geração de relatórios. Assim, para o protótipo proposto, foi alterada a fonte de dados que alimenta a aplicação, além da inserção do contexto colaborativo e a utilização do padrão de projeto *observer* como solução para gerenciar as dependências entre os objetos do relatório. Além disso, a possibilidade de o usuário poder importar diversas estruturas de dados para ferramenta torna a proposta interessante por não atender um domínio específico, possibilitando à ferramenta atender diversos nichos de mercado.

Almeida (2011) destaca que, dentre os componentes para criação dos relatórios, os que apresentam seus conteúdos inseridos dinamicamente são:

- **Tabela**: componente composto por linhas e colunas, podendo ter título e/ou legenda;
- **Gráfico**: os tipos de gráfico podem variar entre pizza, barra ou coluna, podendo apresentar legenda ou não;
- **Valor Escalar**: representa o retorno de um valor único, podendo ser numérico ou textual. Caso seja numérico o usuário poderá escolher algumas opções de agregação, são elas: soma, total, maior valor e menor valor.

Todos os componentes citados anteriormente têm seus controles divididos em três níveis: Propriedades, Configurações e Filtros. Como já citado, esses componentes foram definidos em Almeida (2011). Para o protótipo desenvolvido no presente trabalho, novos conceitos foram agregados à ferramenta e serão melhor detalhados na Figura 18.

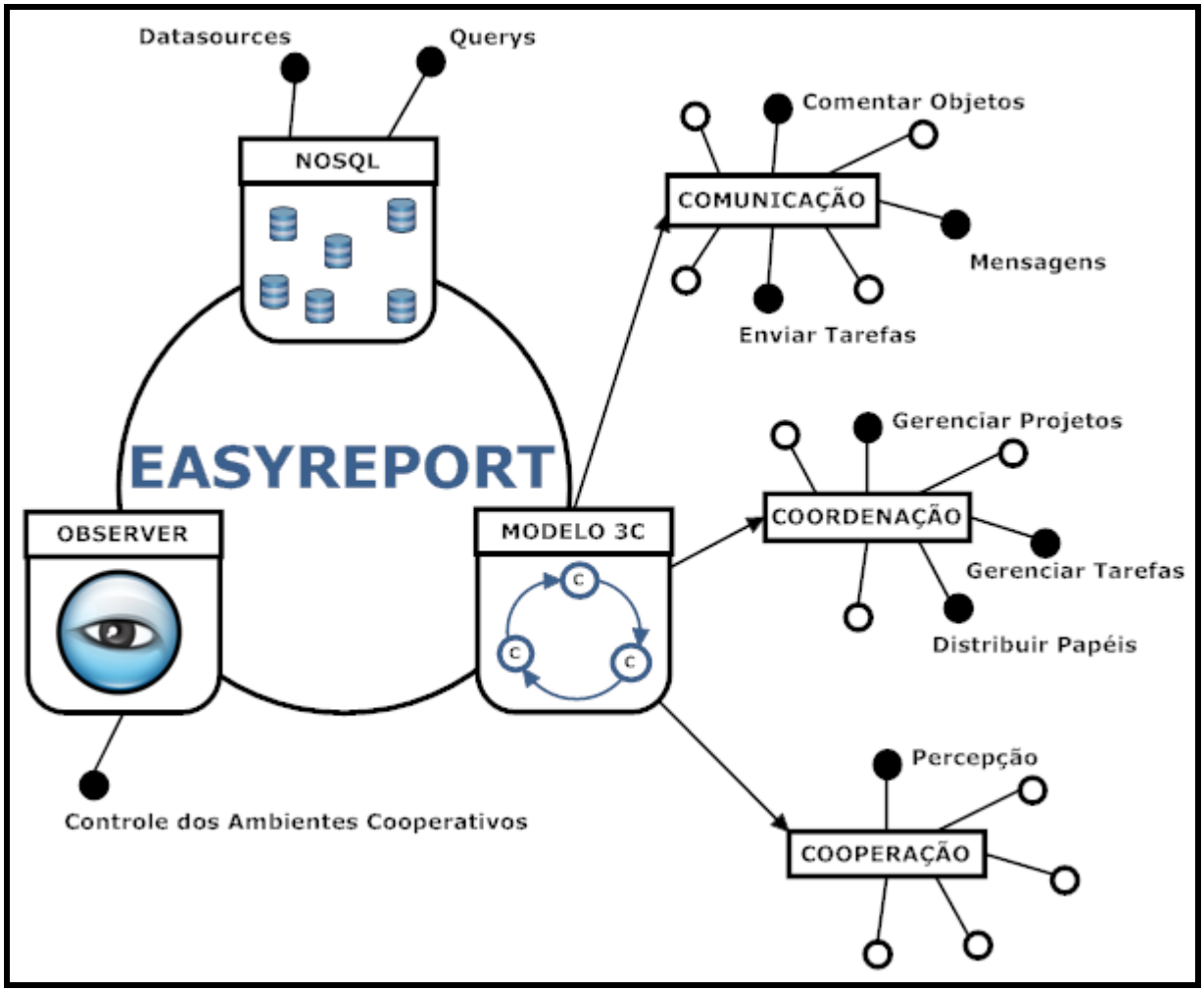


Figura 18 - Escopo do Protótipo

Conforme apresentado na Figura 18, o protótipo foi desenvolvido em três principais soluções e tem seu escopo da seguinte forma:

- ***Observer*** – este padrão de projeto prevê solucionar questões de dependências entre os objetos distribuídos em diversas estações de trabalho que trabalham em um mesmo ambiente de cooperação. No protótipo desenvolvido o padrão foi aplicado no:
    - **Controle dos Ambientes Cooperativos** – para exemplificar o papel do padrão de projeto *observer* no protótipo, será apresentado o seguinte cenário: caso o "Usuário A" esteja trabalhando no mesmo ambiente cooperativo que o "Usuário B", as alterações realizadas pelo "Usuário A" devem refletir na interface do "Usuário B", em que a recíproca é verdadeira, assim, o padrão de projeto *observer* é encarregado de gerenciar as dependências entre os objetos de interface da ferramenta;

- **NOSQL** – para o desenvolvimento do protótipo foi definido que será oferecido suporte inicial para importação de dados advindos do *Microsoft SQL Server*, por ser suficiente para demonstrar a aplicação do NOSQL como solução. O protótipo tem como um dos objetivos específicos apresentar a utilização da classe de banco de dados NOSQL como solução para flexibilidade de armazenamento de diversas estruturas de dados, alta escalabilidade e disponibilidade. O desenvolvimento das soluções NOSQL foi dividido em dois pontos distintos, são eles:
    - ***Datasources* –** o usuário pode importar diversas bases de dados *SQL Server* separadamente, formando assim um repositório de *datasources* (fonte de dados);
    - ***Querys* –** o usuário tem ferramentas para construir filtros com propósito de estabelecer os dados que serão apresentados em um determinado objeto (como um gráfico, por exemplo), essas consultas são ser montadas dinamicamente pelo protótipo.
- **Modelo 3C** – o protótipo conta apenas com os *groupwares* necessários para demonstrar a utilização do padrão de projeto *observer* e a classe de banco de dados NOSQL como soluções. Os *groupwares* desenvolvidos foram categorizados nas dimensões do Modelo 3C e são listados a seguir:
    - **Cooperação** – dentre as diversas possibilidades de elementos de cooperação, o protótipo foca no ambiente cooperativo, em que diversos usuários podem trabalhar com os mesmos objetos de forma colaborativa, dando foco em elementos de percepção (bloquear um objeto enquanto está sendo editado, informar quem está editando o objeto, indicar novas mensagens sobre o objeto, etc.);
    - **Comunicação** – o coordenador pode enviar mensagens para estabelecer relações de compromisso sobre novas tarefas (podendo haver troca de mensagens para renegociações), além disso, os usuários podem trocar informações em cada objeto do relatório e também mensagens síncronas no projeto;
    - **Coordenação** – o coordenador pode gerenciar os projetos, criando novos, cadastrando *datasources* e vinculando os colaboradores. Há também um gerenciador de tarefas, para as tarefas iniciadas através da ferramenta de comunicação, após o compromisso ter sido assumido, o coordenador pode acompanhar o andamento da tarefa.

A próxima seção apresenta os requisitos funcionais do sistema, que abrangem as ações que o sistema permitirá para o usuário.

4.2 REQUISITOS FUNCIONAIS

Os Requisitos Funcionais são importantes para estabelecer um entendimento das operações que serão desenvolvidas para o sistema. O desenvolvimento dos requisitos foi realizado tendo a partir de especificações definidas como básicas para o trabalho colaborativo na construção dos relatórios. A seguir são apresentados os requisitos funcionais do sistema:

- Requisitos já definidos na versão anterior da ferramenta (mais detalhes em Almeida (2011)):
    - **Gerenciar Tabela**: Inserir tabela; Posicionar tabela; Definir legenda; Definir título; Editar estilo da tabela; Remover tabela;
    - **Gerenciar Gráfico**: Inserir gráfico; Posicionar gráfico; Definir legenda; Definir título; Editar tipo de gráfico; Remover gráfico;
    - **Gerenciar Valor Escalar**: Inserir totalizador; Posicionar totalizador; Editar estilo do valor escalar; Remover valor escalar;
    - **Gerenciar Área de Texto**: Inserir área de texto; Posicionar área de texto; Editar estilo do texto; Remover área de texto;
    - **Gerenciar Cabeçalho**: Definir cabeçalho; Preencher cabeçalho;
    - **Gerenciar Rodapé**: Definir rodapé; Preencher rodapé; Habilitar paginação;
    - Definir Filtro;
    - Definir Configuração;
    - **Gerenciar Dados**: Manipular relatório; Inserir informações do relatório; Pré-visualizar; Gerar Relatório.
- Novos requisitos propostos para o protótipo de ferramenta:
    - **Cadastrar Relatórios –** O coordenador poderá cadastrar novos relatórios, indicando os colaboradores;
    - **Distribuir papéis –** No momento em que o coordenador estiver cadastrando ou editando os dados iniciais de um relatório, ele poderá vincular os colaboradores e indicar qual o perfil de cada um dentro do projeto, os possíveis perfis são:
        - **Coordenador –** Mesmas permissões que o coordenador que iniciou o projeto, além de poder alterar as informações iniciais do projeto, vincular novos integrantes e indicar os perfis, todo coordenador (inclusive o que criou o projeto) tem papel de colaborador;

    - **Colaborador –** Não tem acesso a distribuição de papéis, tarefas e edição dos dados iniciais do projeto, mas é responsável por trabalhar no ambiente colaborativo com os demais colaboradores;
    - **Visualizador –** Só pode acompanhar os trabalhos e se comunicar com a equipe, não interage com os objetos contidos no ambiente de cooperação.
- **Importar Dados –** O coordenador poderá importar diversos arquivos CVS, em que cada um será cadastrado como um *datasource* separadamente;
- **Selecionar *Datasources* –** Cada objeto dentro de um relatório poderá ser relacionado a um *datasource* diferente, por exemplo, em um gráfico pode-se vincular um *datasource* com dados de uma instituição de ensino de São Paulo, enquanto em outra tabela pertencente ao mesmo relatório o usuário poderá vincular outro *datasource* referente aos dados de uma instituição de ensino do Tocantins;
- **Bloquear objetos em processo de edição –** No momento que o "Usuário A" mudar o estado de um objeto para "edição", este objeto será bloqueado para os demais usuários, estes usuários terão informações de qual usuário está editando, além disto, o objeto ficará destacado (borda realçada, por exemplo) assim que entrar no processo de edição, voltando ao normal no termino da edição;
- **Comentar objetos –** Todo colaborador poderá comentar informações exclusivas de um objeto específico, por exemplo, em um gráfico o "Usuário A" abre a telas de comentários do gráfico em questão e faz um questionamento, imediatamente o "Usuário B" é informado sobre uma nova mensagem no gráfico, visualiza e responde;
- **Distribuir tarefas –** O coordenador, através de uma ferramenta de comunicação, envia mensagens específicas para um colaborador a fim de lhe indicar uma tarefa. Junto à mensagem de texto serão inseridos símbolos para indicar o estado da tarefa, no decorrer da execução da tarefa tanto o coordenador quanto os colaboradores da tarefa poderão atualizar seu estado. Além disso, o usuário que receber a tarefa pode abrir um canal de comunicação e renegociar com o coordenador;
- **Trocar Mensagens Síncronas –** Os coordenadores, colaboradores e visualizadores irão compartilhar de um mesmo espaço para cada projeto, onde poderão trocar mensagens de forma síncrona, a fim de melhorar a comunicação e evitar possíveis problemas no processo de colaboração.

A próxima seção apresenta o papel do Modelo 3C de colaboração no protótipo.

4.3 APLICAÇÃO DO MODELO 3C DE COLABORAÇÃO NO PROTÓTIPO

O Modelo 3C de colaboração foi a abordagem escolhida para a ferramenta proposta, com objetivo de se definir os *groupwares* necessários para uma ferramenta colaborativa para criação de relatórios. Porém, deve-se considerar que o trabalho se trata de um protótipo, cujo objetivo principal é demonstrar a utilização da classe de banco de dados NOSQL como solução de escalabilidade e flexibilidade para aceitar qualquer tipo de estrutura de dados, além do padrão de projeto *observer* para gerenciar os objetos do ambiente de cooperação que apresentam dependências entre si. Logo, referente ao Modelo 3C de colaboração, apesar da ferramenta ter *groupwares* referentes às três dimensões do modelo (conforme apresentado no escopo), o foco está centrado na dimensão de cooperação, inserindo elementos de percepção para controle e auxílio do trabalho colaborativo. Os elementos trabalhados são:

- **Comentários dentro de objetos (Comunicação)** – para cada objeto dentro do relatório (como gráficos, tabelas, valores escalares), os usuários poderão inserir comentários específicos, no momento que isso ocorrer um sinal visual será gerado na tela e todos os usuários ficarão cientes que há um novo comentário para o objeto;
- **Tarefas (Coordenação)** – quando um coordenador envia ou renegocia uma tarefa os usuários encarregados da tarefa são informados através de um elemento de percepção.
- **Mensagens enviadas (Comunicação)** – sempre que uma nova mensagem for enviada por um usuário todos serão informados através de um elemento visual;
- **Objetos em processo de edição (Cooperação)** – esse é o elemento mais importante dos trabalhados para o protótipo, sempre que um objeto entrar em processo de edição, o objeto será bloqueado, e os outros usuários visualizarão esse bloqueio por uma alteração visual, também será informado para todos os usuários qual usuário está editando determinado objeto.

A próxima seção apresenta o papel da classe de banco de dados NOSQL no protótipo.

4.4 APLICAÇÃO DA CLASSE DE BANCO DE DADOS NOSQL NO PROTÓTIPO

O fato de a ferramenta poder aceitar qualquer estrutura de dados descarta a utilização do modelo relacional de banco de dados, pois estes apresentam estrutura rígida. Uma possível

solução demandaria a criação de um modelo abstrato, que poderia ter muito custo para aplicação. Além disso, aceitar qualquer estrutura de dados deixa a aplicação sem a certeza do volume de dados que o usuário irá importar, o que pode trazer problemas de escalabilidade, em que o aumento do volume de dados afeta diretamente na performance da ferramenta (que pode atrapalhar o processo de colaboração). Para solucionar tanto a flexibilidade de aceitar qualquer estrutura de dados quanto para promover alta escalabilidade, a classe de banco de dados NOSQL foi utilizada para construção do protótipo.

Dentre os principais tipos de banco de dados NOSQL, para a ferramenta proposta neste trabalho, foi escolhido o banco de dados orientado a documentos, pois não apresenta uma estrutura pré-definida, podendo ser alterada a qualquer momento, além de oferecer alta escalabilidade e disponibilidade. O banco de dados orientado a documentos escolhido foi o MongoDB, que oferece vários dos recursos avançados de consulta (como filtragem, agregação e classificação), necessários para a construção dos filtros dinâmicos previstos para ferramenta. A Figura 19 demonstra como a solução NOSQL funcionará no protótipo.

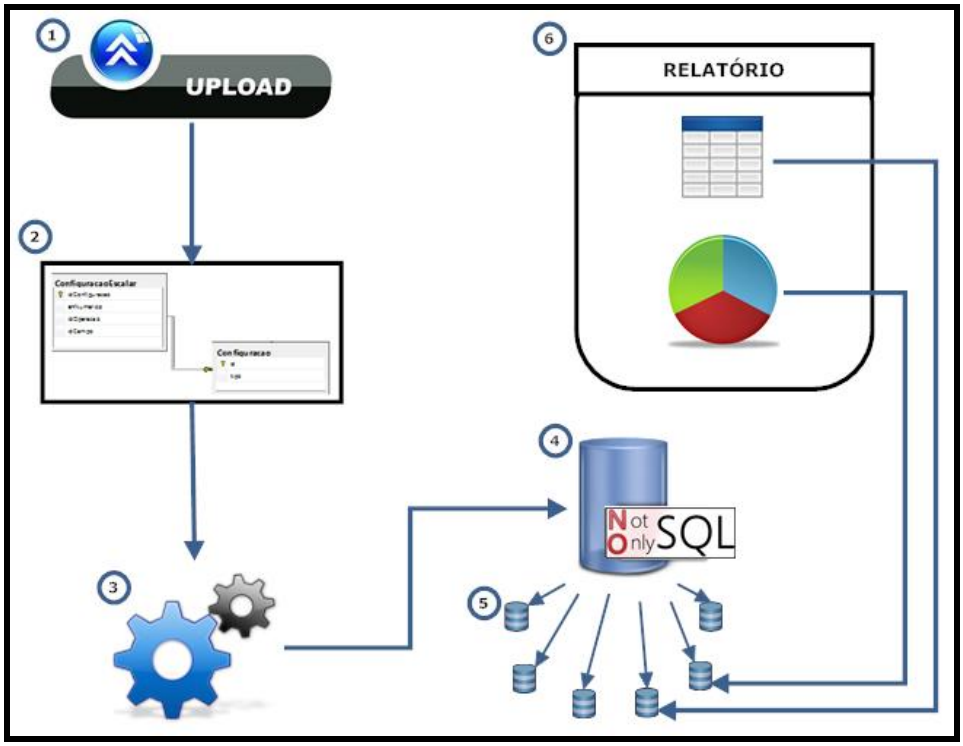


Figura 19 - Papel do NOSQL no protótipo

A partir do esquema da Figura 19, podem-se observar os seguintes passos na utilização da classe de banco de dados NOSQL:

- **Item 1** – representa o momento em que o usuário indica a base de dados que deseja importar;

- **Item 2** – a ferramenta se conecta a base de dados indicada pelo usuário;
- **Item 3** – através de um mecanismo escrito na linguagem PHP, a estrutura dos dados importados pelo usuário é convertida do modelo relacional para o modelo NOSQL. Por fim, após criar a estrutura de documento os dados são migrados para base NOSQL;
- **Item 4** – indica o banco de dados que guarda todas as fontes de dados importadas pelo usuário;
- **Item 5** – representam os *datasources* gerados a partir das importações. Para que a ferramenta diferencie cada importação criada pelo usuário, é inserido a cada documento o nome do *datasource* a que ele pertence, este dado é posto em cada consulta montada dinamicamente na ferramenta;
- **Item 6** – é um exemplo de relatório final, que contem dois objetos, um gráfico e uma tabela, em que cada objeto foi gerado a partir de um *datasource* diferente. Por exemplo, o gráfico pode estar utilizando um *datasource* referente à dados de uma instituição de ensino de São Paulo, enquanto a tabela está sendo preenchida por dados advindos de um *datasource* com informações de uma instituição de ensino do Tocantins;

Na próxima seção será apresentado o papel do padrão de projeto *observer* no protótipo.

4.5 APLICAÇÃO DO PADRÃO DE PROJETO OBSERVER NA FERRAMENTA

O padrão de projeto *observer* dentro do protótipo tem a função de controlar as dependências entre os objetos de um mesmo projeto de relatório. Além disso, o padrão facilitará o processo de manutenção da ferramenta colaborativa, considerando que ferramentas colaborativas estão susceptíveis a consecutivas atualizações e agregação de novas ferramentas. O padrão permite que novos objetos sejam inseridos na ferramenta (como índice e paginação, por exemplo) sem que o controle de dependências realizado pela classe *Subject* seja afetado. A Figura 20 demonstra como o padrão *observer* funcionará no protótipo.

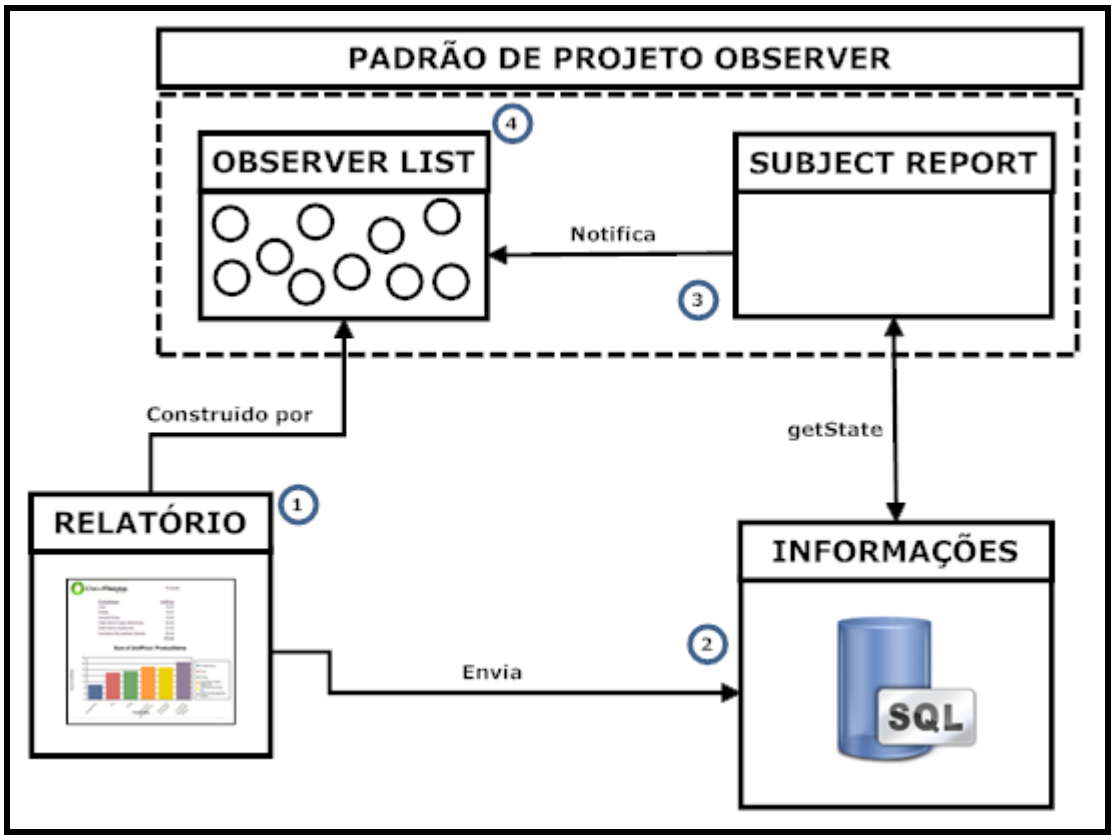


Figura 20 - Aplicação do Padrão de Projeto *Observer* no Protótipo

A partir do esquema da Figura 20, podem-se observar os seguintes passos na utilização do padrão de projeto o*bserver*:

- **Passo 1** – o usuário realiza uma alteração no relatório, mudando, por exemplo, a posição de um gráfico;
- **Passo 2** – a nova posição do gráfico é enviada para o banco de dados relacional que armazena a estrutura dos relatórios;
- **Passo 3** – em todas as estações de trabalho que estão manipulando o mesmo relatório, existe uma classe *Subject* do lado do cliente (implementada na linguagem *ActionScript*), que como um serviço contínuo fica verificando a base de dados (de forma assíncrona, através da função *getState*). O *Subject* recupera a alteração da posição do gráfico e compara com seu estado, verificando a alteração de estado ele notifica sua lista de *observers*;
- **Passo 4** – todos os objetos de um relatório (como gráfico, tabela, rodapé) são implementações da interface *Observer*, estas implementações são notificadas pelo *Subject* e realizam *update* de seus estados quando necessário.

Na próxima seção será apresentada a arquitetura do sistema, que tem por objetivo demonstrar a sequência de passos realizados internamente pelas principais operações do sistema.

4.6 ARQUITETURA DO SISTEMA

A Arquitetura do Sistema é importante para deixar clara a estrutura lógica dos processos realizados, desde a interação entre os usuários na construção do relatório até a emissão em formato de arquivo. A Figura 21 apresenta a versão final da Arquitetura do Sistema.

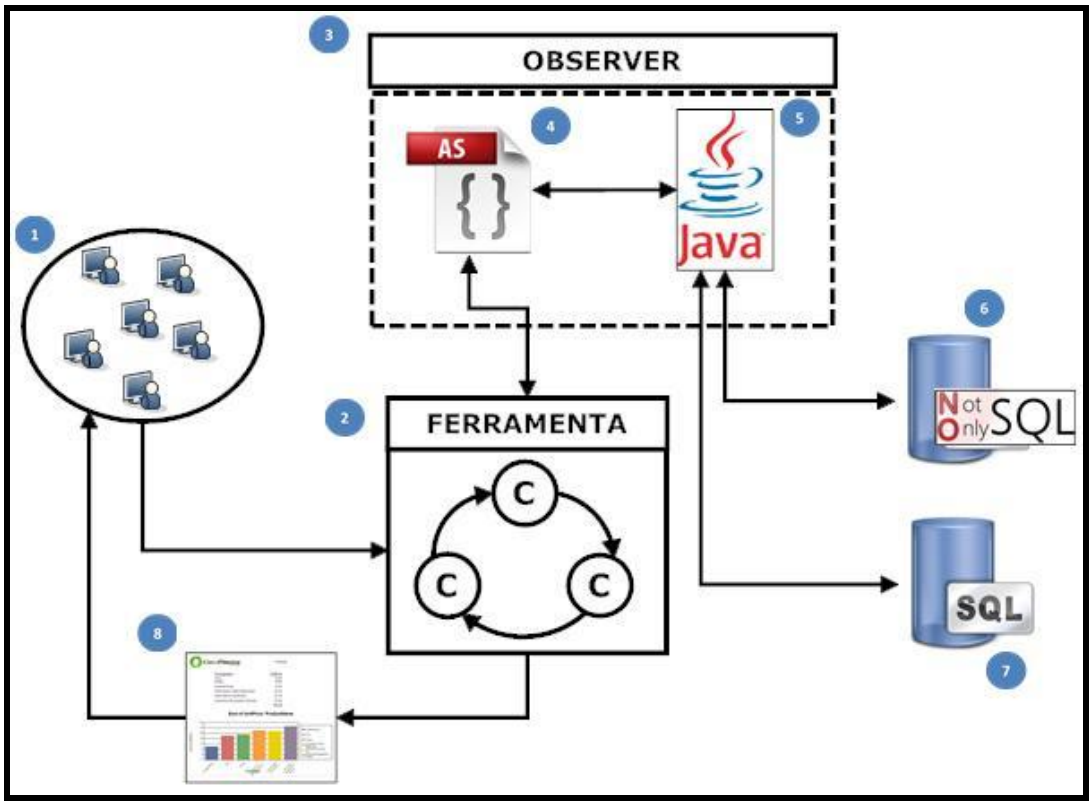


Figura 21 – Arquitetura da Ferramenta Para Criação de Relatórios

A partir do esquema da Figura 21, pode-se observar a seguinte estrutura:

- **Item 1** – representa o grupo de usuários que trabalham de forma colaborativa em um mesmo ambiente compartilhado (Item 2), com objetivo comum de gerar ao final um relatório (Item 8) com os dados armazenados na ferramenta (contidos no Item 6).
- **Item 2** – indica a ferramenta colaborativa baseada no Modelo 3C de Colaboração, em seu espaço de cooperação disponibiliza de recursos "*drag and drop*", além de acesso a qualquer estrutura de dados armazenados na base de dados NOSQL (Item 6).

- **Item 3** – são *scripts* desenvolvidos através da linguagem *ActionScript*, que permitem a manipulação da interface gráfica (Item 2) e a comunicação síncrona com a camada de negócio (Item 4);
- **Item 4** – é a camada de negócio responsável por armazenar e recuperar os dados contidos na base de dados da ferramenta (Item 7), que inclui os colaboradores, os projetos, as estruturas dos relatórios cadastrados pelos usuários (Item 1) etc.. Esta camada também é responsável por importar e acessar as informações obtidas através de fontes de dados externas indicadas pelos usuários (as informações são armazenadas na base de dados NOSQL Item 6).
- **Item 5** – representa o padrão de projeto *observer* que será utilizado para concepção da ferramenta e afetará tanto a camada de apresentação (Item 3) quanto a camada lógica de negócio (Item 4).
- **Item 6** – base de dados NOSQL que armazenará os dados importados pelos usuários (Item 1). Poderão ser importadas diversas estruturas de dados, sem que uma interfira nas outras, isso permitirá que o usuário possa escolher "N" fontes de dados para um mesmo projeto.
- **Item 7** – base de dados relacional da ferramenta que armazenará os modelos de relatórios criados pelos usuários (Item 1).
- **Item 8** – representa o relatório gerado a partir da estrutura de relatório criada pelo usuário.

Na próxima seção será apresentado o *storyboard* da ferramenta, que tem por objetivo esboçar as características das telas do sistema.

4.7 STORYBOARD

O *storyboard* foi desenvolvido com objetivo de apresentar uma visão detalhada das funcionalidades contidas no protótipo. Algumas telas foram definidas na versão anterior da ferramenta e, por isto, não serão apresentadas, mas podem ser vistas com mais detalhes em Almeida (2011), são elas:

- **Visualização geral da ferramenta** – apresenta as formas de acesso a ferramentas e a recursos contidos no ambiente de trabalho (como lista de objetos, área de trabalho, lista de relatórios etc.).
- **Cadastro de relatório** – define os campos de cadastro em um relatório e como são organizados em estrutura de árvore.
- **Barra de controle** – um facilitador de acesso as funcionalidade de um objeto do relatório.

- **Propriedades do objeto** – são informações sobre características visuais dos objetos (por exemplo, altura, largura, cor do título etc.).
- **Filtro do objeto** – apresenta a listagem de filtros cadastrados para o objeto vigente, estes filtros podem ser cadastrados, editados ou deletados a qualquer momento.
- **Configuração da tabela** – contêm os campos disponíveis para relacionar ao objeto.
- **Configuração do gráfico** – armazena informações, como: se é horizontal, vertical, se deve apresentar ou não legenda;
- **Configuração valor escalar** – representa o campo que deverá ser apresentado, caso o campo seja numérico pode-se escolher o tipo de operação (soma, contador etc.)
- **Exemplo de produto final –** é uma demonstração de um relatório em sua versão final.

A Figura 22 exibe a funcionalidade referente à importação de dados advindos de fontes de dados (*SQL Server*) externas.

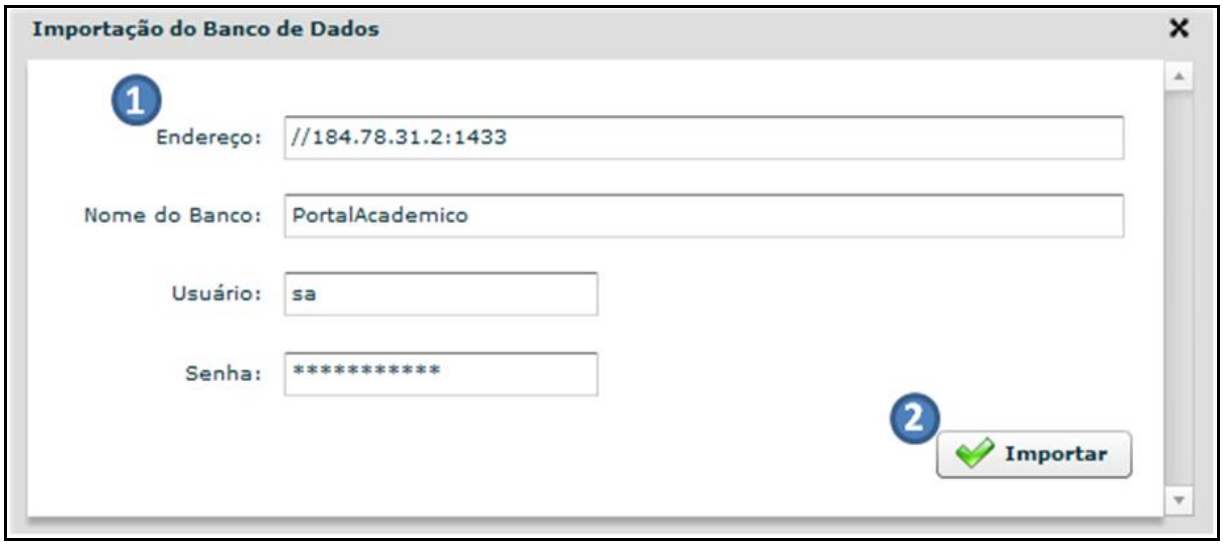


Figura 22 - Importação do Banco de Dados

Pode-se observar a partir da Figura 22 a maneira com que o usuário irá informar a base de dados relacional que deseja importar para ferramenta. Primeiramente, o usuário abrirá a *pop-up* e preencherá um formulário (Item 1) contendo os dados de acesso do banco de dados relacional que deseja importar. Após isso, o usuário confirmará a ação através do botão "Importar" (Item 2). Assim, conforme pode ser observado na Figura 23, no momento em que o botão for acionado, a ferramenta utilizará os dados de acesso informados (Item 1) para acessar a estrutura da base e converter o modelo relacional para o modelo orientado a documentos, importando os dados ao final da operação. Cada base de dados importada pelo

usuário gerará um novo *datasource* que poderá ser relacionado a um objeto contido em um relatório.

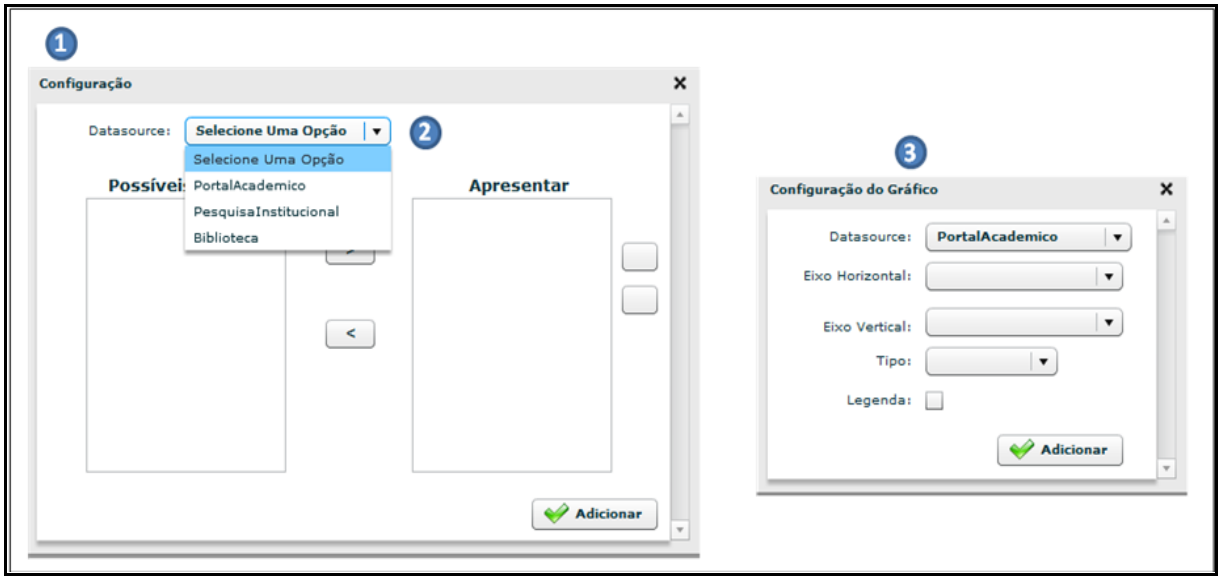


Figura 23 – Seleção de um datasource para um objeto

Na Figura 23, pode-se verificar que, para cada tipo de objeto que pode carregar valores de forma dinâmica (como tabelas, gráficos, valores escalares), na tela de configuração foi acrescentado um elemento *combobox* (Item 2) que lista os *datasources* disponíveis para o objeto que está sendo editado. A tela de configuração de uma tabela, além do *combobox* que lista os *datasource* disponíveis (Item 2), também apresenta um espaço que permite ao usuário informar os campos que deseja apresentar na tabela. Os campos disponíveis são apresentados assim que o usuário seleciona um *datasource*, até que isto ocorra, a listagem de possíveis campos para apresentação fica desabilitada. O mesmo ocorre com a *pop-up* de configuração de um gráfico (Item 3), em que só após o usuário informar o *datasource*, a ferramenta apresenta os possíveis campos para definir o "Eixo Horizontal" e o "Eixo Vertical" (Item 3). A Figura 24 apresenta o controle de permissões que define como os usuários poderão interagir em um relatório.

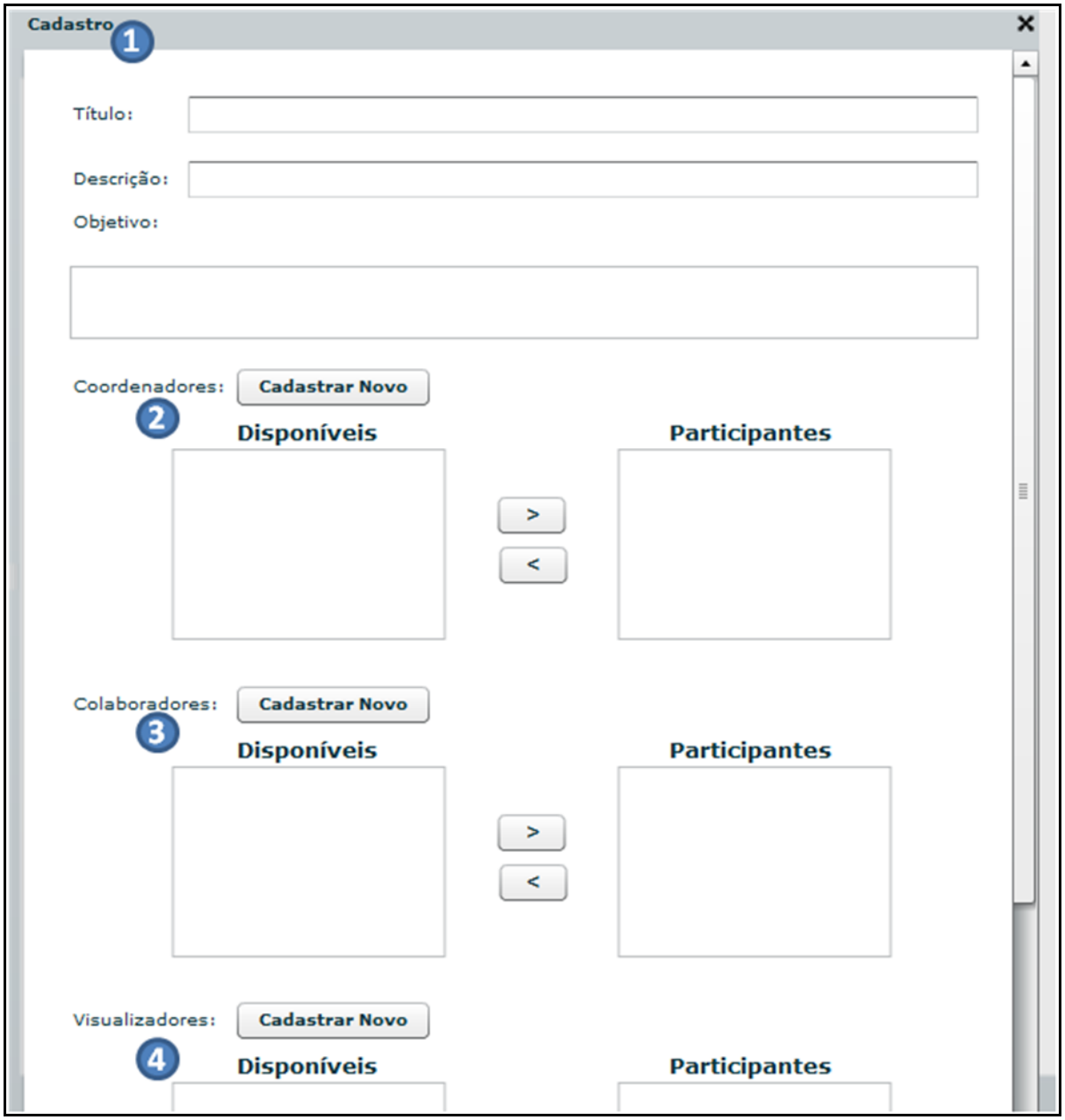


Figura 24 – Controle dos papéis para um relatório

A Figura 24 apresenta a tela de cadastro e edição de relatórios (Item 1). Essa tela já havia sido definida em Almeida (2011), entretanto, foi acrescentado o controle de papéis para o relatório trabalhado. O coordenador pode, através da tela (Figura 24), relacionar usuários com perfil de: coordenador, colaborador ou visualizador (Itens 2, 3 e 4 respectivamente). Para as três opções de perfis, caso a pessoa relacionada não esteja cadastrada no sistema, o coordenador poderá cadastrá-la. Por exemplo, caso o coordenador cadastre o "Usuário X" através do botão "Cadastrar Novo", que está na região da relação de colaboradores (Item 3), o "Usuário X", após cadastrado, aparecerá nas três listagens (Itens 2, 3 e 4) como disponível. Então, assim que o coordenador passar o "Usuário X" para listagem de colaboradores

participantes, o “Usuário X” não estará mais disponível para nenhum dos três papéis, pois já estará relacionado a um deles. A Figura 25 apresenta a forma com que o coordenador controla as tarefas atribuídas aos colaboradores de um relatório.

Figura 25 - Gerenciador de tarefas

A tela que gerencia uma tarefa (Figura 25) é constituída de dados iniciais de cadastro, que são: título da tarefa, sua descrição, situação e porcentagem de conclusão da tarefa (Item 1). Ainda nesta tela, o coordenador pode relacionar os colaboradores que serão responsáveis por determinada tarefa (Item 2), além de ter a opção, a qualquer momento da execução da

tarefa, de inserir ou excluir colaboradores (Item 2). No momento em que um coordenador envia uma tarefa para um ou mais colaboradores, os colaboradores tem acesso a nova tarefa em sua lista de tarefas e, ao acessar a nova tarefa visualizam uma mensagem com informações de abertura da tarefa (como apresentado no Item 3-A), em que o coordenador enviou a mensagem “Iniciem essa tarefa o quanto antes”. O coordenador envia a mensagem com objetivo dos colaboradores assumirem o compromisso de execução da tarefa, entretanto o gerenciador de tarefas permite que seja criado um canal de comunicação entre os colaboradores e o coordenador que criou a tarefa. No item 3-B, um colaborador realiza um questionamento quanto a composição do gráfico solicitado pelo coordenador. Além disso, toda mensagem relacionada a uma tarefa apresenta um contexto que está relacionado a um símbolo (Item 4). Assim, na situação apresentada na Figura 25, o usuário abre um diálogo para renegociar a tarefa. A Figura 26 apresenta o canal de comunicação relacionado ao relatório que está sendo trabalhado.

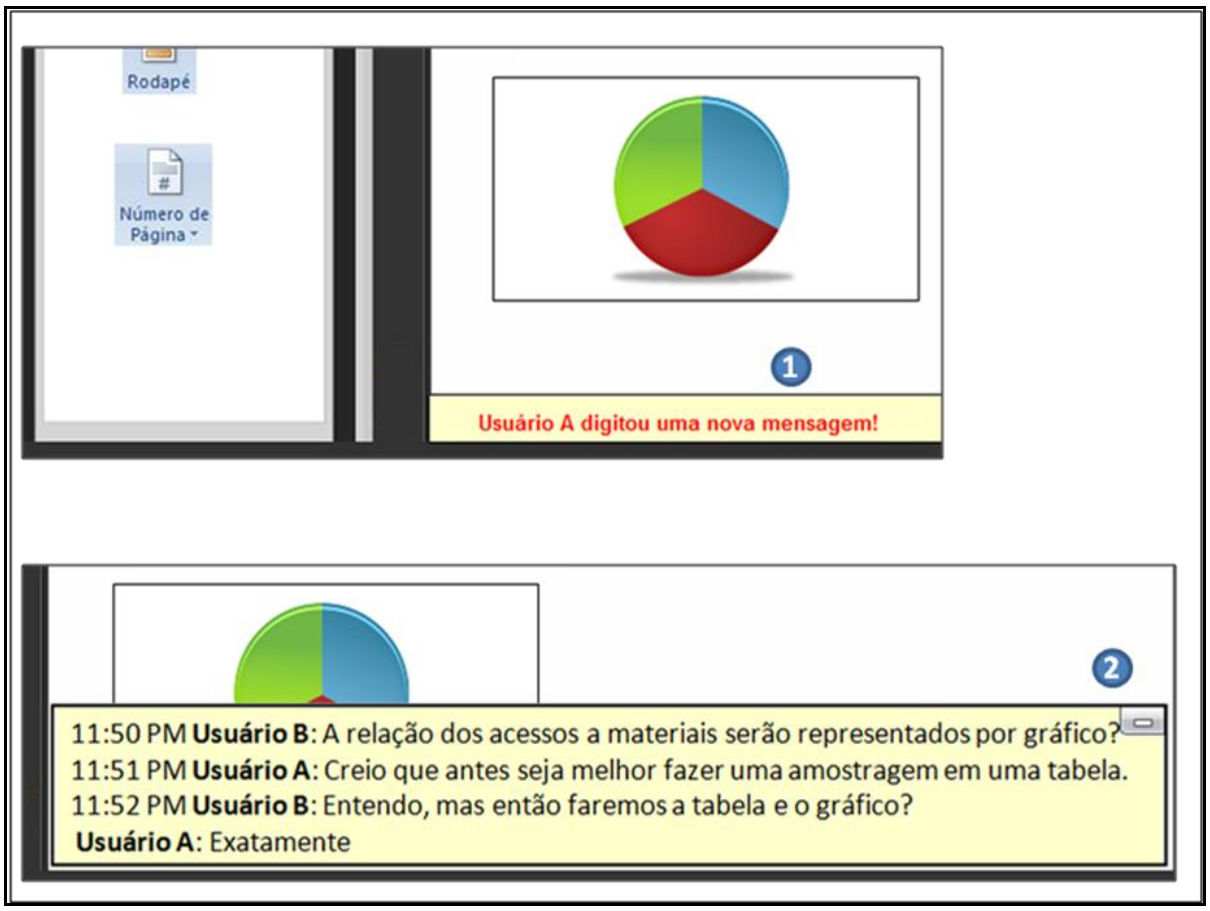


Figura 26 - Comunicador geral do relatório

Pode-se observar, a partir da Figura 26, que todo relatório cadastrado oferece um canal de comunicação que não apresenta um contexto específico, não estando relacionada especificamente a um objeto do relatório ou a uma tarefa, por exemplo. Caso o "Usuário A" envie uma nova mensagem para o grupo, todos serão informados através de uma mensagem na extremidade inferior da área de trabalho (Item 1). Selecionando o alerta de nova mensagem, os usuários poderão visualizar a nova mensagem enviada pelo "Usuário A" (Item 2). Esse canal de comunicação visa discussões gerais referentes ao relatório trabalhado, em que o histórico de conversa é mantido, informando o autor das mensagens e os horários que as mensagens foram enviadas, a fim de que cada participante do relatório possa recuperar informações como decisões realizadas em grupo. Além desse canal de comunicação, que tem contexto mais abrangente, há também o canal de comunicação referente a um objeto específico do relatório, apresentado na Figura 27.

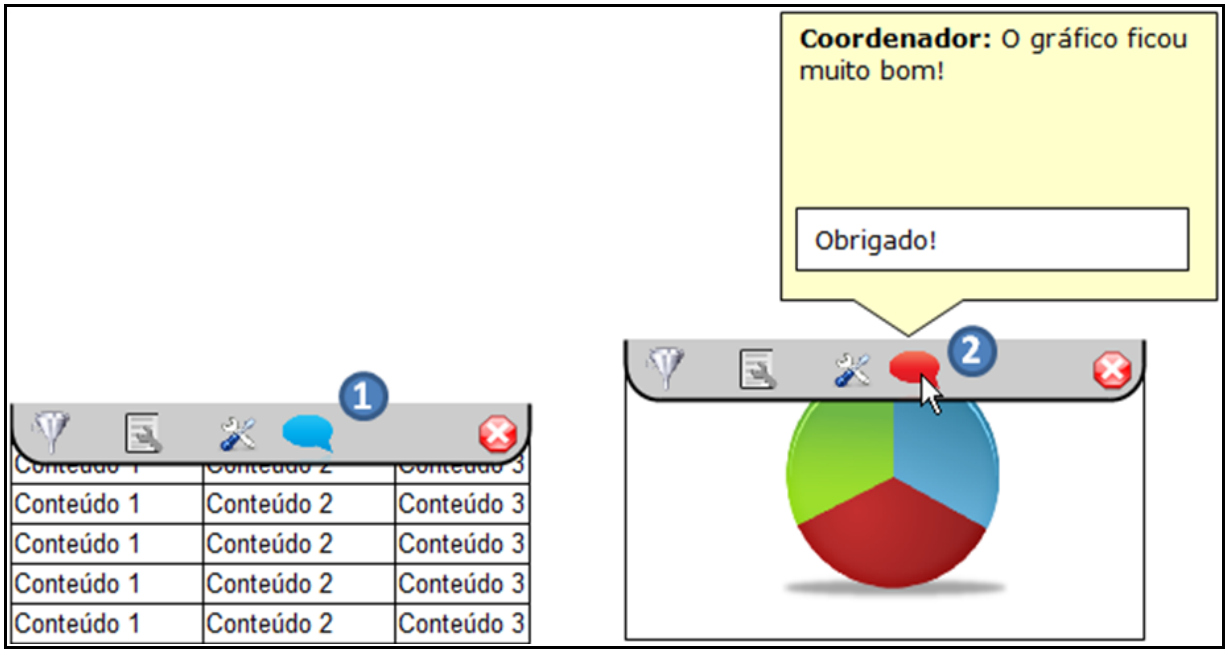


Figura 27 - Comunicador do objeto

Pode-se notar, na Figura 27, que cada objeto de um relatório apresenta opção para troca de mensagens sobre o objeto (Itens 1 e 2). Quando não há novas mensagens de um determinado objeto, é apresentado um balão azul (Item 1). Porém, quando algum usuário envia uma nova mensagem, todos os outros usuários são informados visualizando um balão vermelho (Item 2). Ao selecionar no balão vermelho, o usuário visualizará o histórico de mensagens e poderá, se necessário, enviar uma nova mensagem para os demais colaboradores, para tanto, deverá digitar a mensagem e pressionar a tecla "*enter*".

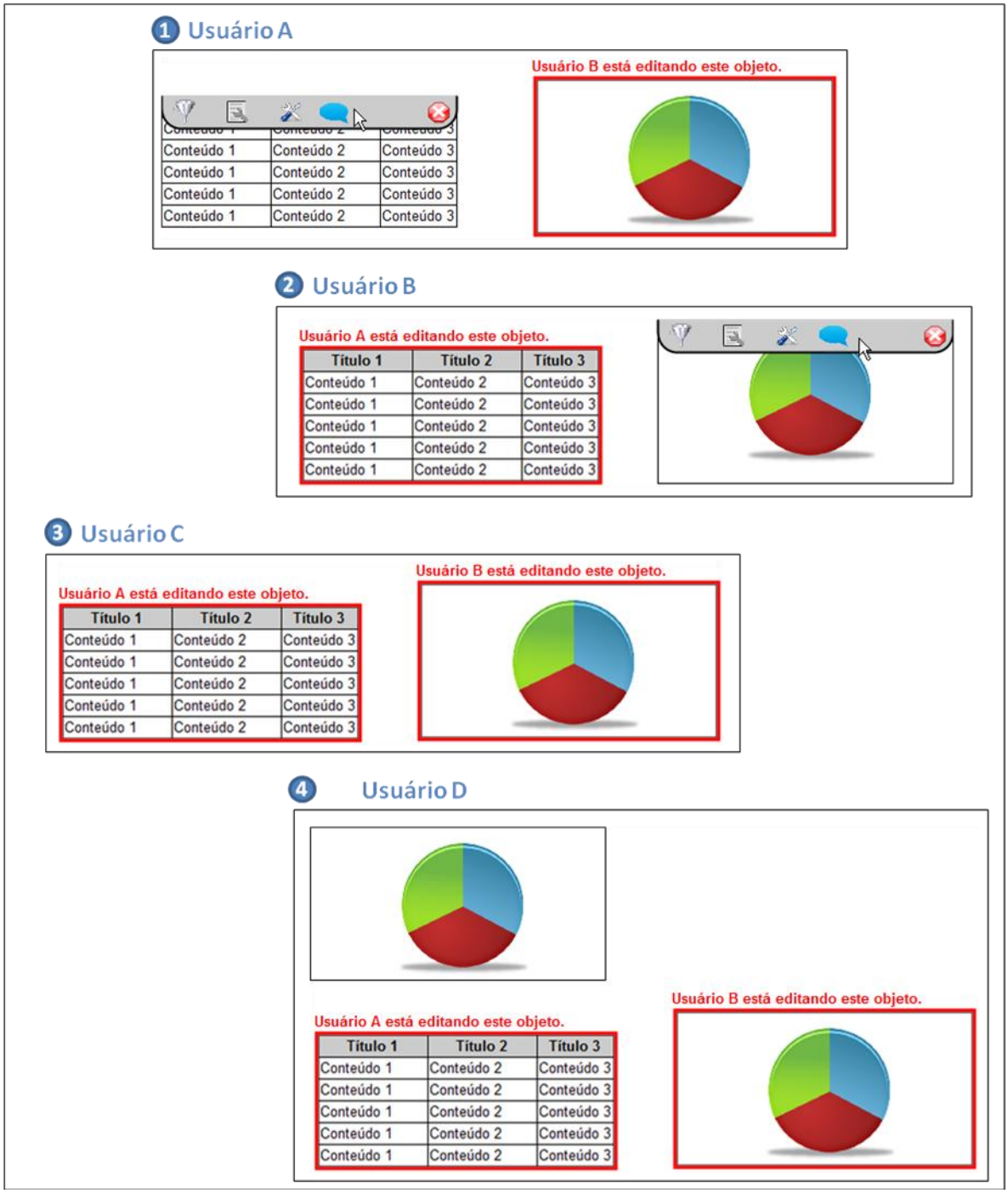


Figura 28 - Elementos de percepção no ambiente de cooperação

A Figura 28 apresenta trechos de quatro áreas de trabalho diferentes a fim de demonstrar como serão gerenciados os elementos de percepção na área cooperativa do protótipo. A seguir é apresentada cada situação dos exemplos apresentados na Figura 28:

- **Item 1** – enquanto o “Usuário A” edita uma tabela, ele também percebe que o gráfico está bloqueado e visualiza uma mensagem que indica que o “Usuário B” está editando o gráfico.
- **Item 2** – o “Usuário B” nota que a tabela que contém o relatório está bloqueada e sendo editada pelo “Usuário A”. Enquanto o “Usuário A” edita a tabela o “Usuário B” edita o gráfico.
- **Item 3** – o “Usuário C” visualiza que tanto a tabela quanto o gráfico estão bloqueados em sua interface. Deseja editar o gráfico do relatório, entretanto, deve esperar até que o Usuário B termine as alterações.
- **Item 4** – o “Usuário D” também percebe que os objetos estão bloqueados (Tabela e Gráfico), entretanto inicia o cadastro de um novo gráfico. Os demais usuários só irão visualizar o novo gráfico quando o “Usuário D” finalizar o processo inicial de cadastro.

Na próxima seção será apresentado o modelo relacional do banco de dados da ferramenta, que tem por objetivo apresentar as particularidades encontradas no modelo lógico.

4.8 MODELO RELACIONAL DO BANCO DE DADOS

Mesmo com os dados analíticos trabalhados de forma colaborativa na ferramenta serem oriundos da classe de banco de dados NOSQL, as estruturas, regras e controles são definidos em um banco de dados relacional. A partir disso, foi desenvolvido o modelo relacional do banco de dados, a fim de demonstrar as relações lógicas entre as entidades e como elas deverão ser trabalhadas. Deve-se destacar que o modelo aplicado na versão anterior da ferramenta foi utilizado para criação do novo modelo. A seguir (Figura 29) é apresentado o modelo relacional da versão anterior da ferramenta, em que a descrição das entidades pode ser encontrada em Almeida (2011).

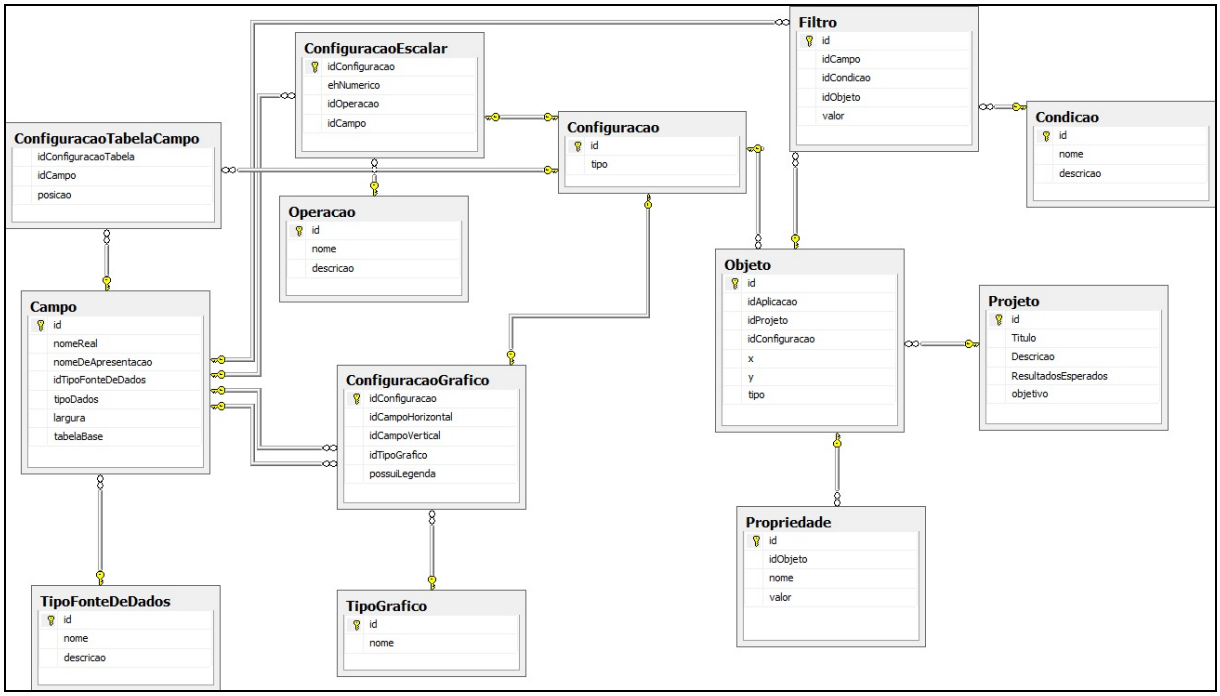
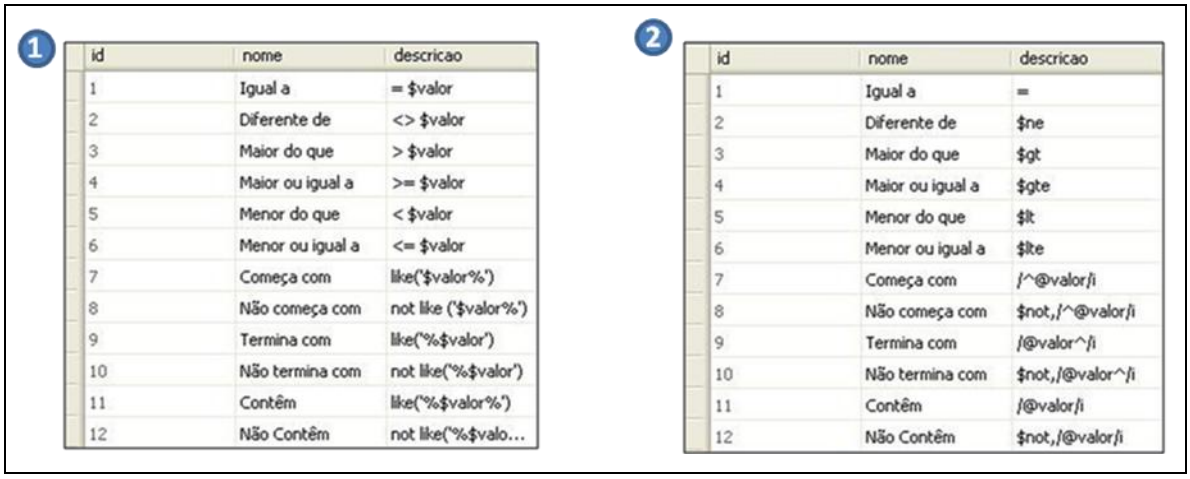


Figura 29 - Diagrama do Banco de Dados (ALMEIDA, 2011, p. 25)

A estrutura estabelecida na Figura 29 foi reaproveitada para o protótipo desenvolvido. Algumas ressalvas devem ser realizadas quanto ao modelo: a entidade “TipoFonteDeDados” manteve seus campos e relacionamentos, porém, teve seu sentido modificado. Anteriormente armazenava as fontes de dados disponíveis para ferramenta que era “base de dados ou *Google Analytics*” (ALMEIDA, 2011, p. 26), agora armazena a referência dos *datasources* cadastrados pelo usuário. Outro detalhe que foi alterado diz respeito ao conteúdo armazenado na tabela “Condicao”, que é uma tabela que armazena as opções de pesquisa permitidas pela ferramenta (como, “maior que”, “começa com”, “igual a”). A seguir (Figura 30) é apresentada a relação do preenchimento da tabela “Condicao” entre o modelo antigo e o novo.



1

id	nome	descricao
1	Igual a	= $valor
2	Diferente de	<> $valor
3	Maior do que	> $valor
4	Maior ou igual a	>= $valor
5	Menor do que	< $valor
6	Menor ou igual a	<= $valor
7	Começa com	like('$valor%')
8	Não começa com	not like ('$valor%')
9	Termina com	like('%$valor')
10	Não termina com	not like('%$valor')
11	Contêm	like('%$valor%')
12	Não Contêm	not like('%$valo...

2

id	nome	descricao
1	Igual a	=
2	Diferente de	$ne
3	Maior do que	$gt
4	Maior ou igual a	$gte
5	Menor do que	$lt
6	Menor ou igual a	$lte
7	Começa com	/^@valor/i
8	Não começa com	$not,/^@valor/i
9	Termina com	/@valor^/i
10	Não termina com	$not,/@valor^/i
11	Contêm	/@valor/i
12	Não Contêm	$not,/@valor/i

Figura 30 - Comparativo de preenchimento da tabela "Condicao"

Pode-se observar, a partir da Figura 30, que todos os tipos de filtragem oferecidos pelo protótipo, tanto em sua versão anterior (Item 1) quanto na versão apresentada no presente trabalho (Item 2), tem as mesmas opções de pesquisa. Na coluna “descricao” do Item 1 é possível notar que as instruções condizem com o padrão SQL e, para cada uma dessas instruções, foi inserido seu equivalente no formato aceito pela API do MongoDB. A Figura 31 apresenta as novas entidades inseridas no modelo relacional do banco de dados.

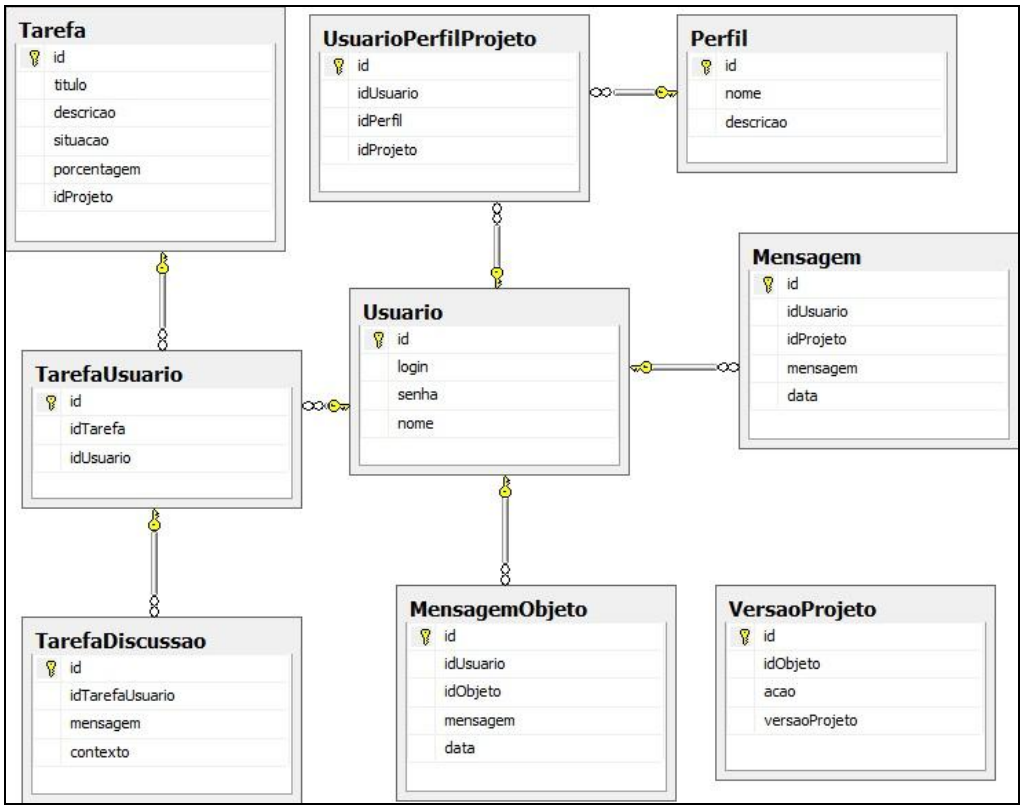


Figura 31 - Novas entidades do modelo relacional do banco de dados

Com base na Figura 31, será realizada uma descrição das novas entidades contidas no modelo relacional do banco de dados:

- "**Usuario**" – representa um indivíduo disponível para ser relacionado a um projeto de relatório.
- "**Perfil**" – armazena os possíveis papéis que um usuário pode assumir, para o protótipo desenvolvido foi estabelecido os seguintes perfis: coordenador, colaborador e visualizador.
- "**UsuarioPerfilProjeto**" – apresenta a relação entre as entidades "Usuario", "Perfil" e "Projeto". Essa relação torna possível um usuário assumir apenas um papel em determinado projeto, porém o usuário pode ter papéis diferentes em projetos diferentes.
- "**Tarefa**" – dados utilizados para definir uma nova tarefa relacionada a um projeto específico.
- "**TarefaUsuario**" – permite que uma tarefa seja alocada para uma lista de usuários, em tempo que o usuário poderá ter mais de uma tarefa.
- "**TarefaDiscussao**" – composta de mensagens trocadas referentes a uma determinada tarefa, para isso se relaciona com a entidade "TarefaUsuario", a partir desta relação é possível estabelecer a tarefa que a mensagem pertence e o usuário responsável pela mensagem enviada.
- "**Mensagem**" – contêm dados de uma mensagem de contexto geral, que aparece na parte inferior do relatório.
- "**MensagemObjeto**" – armazena dados referentes a mensagens de um objeto específico, por exemplo, através desta entidade é possível abrir um discussão entre a equipe para definir os dados apresentados em uma determinada tabela do relatório.
- "**VersaoProjeto**" – permite que seja relacionado o número da nova versão ao objeto modificado, caso mais de um objeto seja alterado para uma mesma versão será gerado um registro para cada objeto, informando qual operação foi efetuado por ele (inserção, *update* ou deleção).

Na próxima seção são apresentados trechos de código que demonstram como o processo de desenvolvimento da ferramenta foi realizado.

4.9 DESENVOLVIMENTO DA FERRAMENTA

O desenvolvimento do protótipo teve como foco a aplicação da classe de banco de dados NOSQL e o padrão de projeto *observer* como soluções para oferecer uma ferramenta colaborativa que permita a construção de relatórios a partir de diversas estruturas de dados. Desta forma, o desenvolvimento da ferramenta pode-se dividir em três áreas principais: importação das bases de dados relacionais, construção dinâmica dos filtros e aplicação do padrão de projeto *observer*. A seguir serão descritos os principais trechos de código para entendimento do funcionamento destas três áreas principais.

4.9.1 Importação de Bancos de Dados Relacionais

Para que os dados de uma base relacional sejam importados em um modelo orientado a documentos é necessário realizar uma série de tratamentos e passos. A seguir é apresentada a função de importação.

```
1  public static function ImportarBase($dadosBaseRelacional)
2  {
3      $retorno = array();
4      $m = new Mongo();
5      $estrutura = gerador::MontarEstruturaParaImportacao($dadosBaseRelacional);
6      $dbm = $m->selectDB($dadosBaseRelacional['nomeBase']);
7      foreach($estrutura as $estruturaEspecifica)
8      {
9          $sql = gerador::MontarSQL($estruturaEspecifica);
10         $retorno = gerador::ExecutaSQL($sql,$estruturaEspecifica,$estrutura);
11         $collection = $dbm->selectCollection($estruturaEspecifica[0]);
12         foreach($retorno as $aux)
13         {
14             $collection->insert($aux);
15         }
16     }
17 }
```

Figura 32 - Importar Banco de Dados Relacional

A função apresentada na Figura 32 recebe como parâmetro do formulário preenchido pelo usuário as informações pertinentes à base de dados relacional que será importada. O primeiro passo para importação é extrair a estrutura da base que está sendo importada. Essa operação é realizada pelo método MontarEstruturaParaImportacao (Linha 5), que retorna um *array* com a relação de tabelas, campos e chaves estrangeiras do banco de dados que será importado. Em seguida, é criado um banco de dados orientado a documentos com mesmo nome da base relacional que será importada (Linha 6). Esta operação é realizada através da

função selectDB (Linha 6), que tem por responsabilidade tanto selecionar bases existentes quanto criar novas bases. Com o retorno da função que monta a estrutura (Linha 5), cada tabela convertida em *array* é percorrida, montando uma consulta para selecionar seus dados (Linha 9) que retorna um *array* com os dados de todos os registros da tabela (Linha 10). Por fim, o retorno dos dados são inseridos no banco de dados MongoDB (Linha 14). A seguir serão apresentadas com detalhes as funções necessárias para importação da base de dados relacional. Primeiramente será apresentada a função que monta a estrutura da base de dados relacional (Figura 33).

```
1  public static function MontarEstruturaParaImportacao()
2  {
3      $res = $db->query("select name as nome from sys.tables");
4      $tabelas = array();
5      while($data = $res->fetchRow())
6      {
7          $tabelas[] = array('nome' => $data->nome);
8      }
9      $columns = array();
10     foreach($tabelas as $tabela)
11     {
12         $nomeTabela = $tabela['nome'];
13         $dbcolumns = $db->query("select nome from sys.columns where".
14                                 "object_id = object_id('".$nomeTabela."')");
15         $sqlForeignKey = "SELECT
16                     COL_nome(fc.parent_object_id,
17                     fc.parent_column_id) AS Columnnome,
18                     OBJECT_nome (f.referenced_object_id) AS ReferenceTablenome,
19                     COL_nome(fc.referenced_object_id,
20                     fc.referenced_column_id) AS ReferenceColumnnome
21                     FROM sys.foreign_keys AS f
22                     INNER JOIN sys.foreign_key_columns AS fc
23                     ON f.OBJECT_ID = fc.constraint_object_id
24                     where OBJECT_nome(f.parent_object_id) = '".$nomeTabela."'";
25         $dbForeignKey = $db->query($sqlForeignKey);
26         $i = 0;
27         $columns[$nomeTabela][0] = $nomeTabela;
28         while($data = $dbcolumns->fetchRow())
29         {
30             $columns[$nomeTabela][1][$i] = $data->nome;
31             $i++;
32         }
33         while($data = $dbForeignKey->fetchRow())
34         {
35             $columns[$nomeTabela][2] = array('columnnome' => $data->Columnnome,
36                                 'referenceTablenome' => $data->ReferenceTablenome,
37                                 'referenceColumnnome' => $data->ReferenceColumnnome);
38         }
39     }
40     return $columns;
41 }
```

Figura 33 - Montar estrutura para importação

No início da função apresentada na Figura 33 (Linha 3) é executada uma consulta que retorna todos os nomes de tabelas da base de dados que serão importados. Em seguida a lista de nomes é percorrida (Linha 10) para consultar as colunas pertencentes a cada tabela (Linha 13). Com intuito de tornar a busca pelas relações entre as tabelas automática é executada uma consulta que retorna as chaves estrangeiras da tabela percorrida, contendo o nome do campo e da tabela que faz relação (Linha 25). A divisão do *array* de cada tabela é realizada da seguinte forma: a primeira posição armazena o nome da tabela (Linha 27), a segunda a lista de colunas (Linha 30) e a terceira a lista de tabelas relacionadas (Linha 35). Ao fim da função que monta a estrutura é retornado (Linha 40) um *array* contendo todos os nomes das tabelas, com suas respectivas colunas e relações com outras tabelas. A função responsável por montar as consultas utilizando como parâmetro os elementos retornados pela função que monta a estrutura é apresentada na figura 34.

```
1   public static function MontarSQL($estrutura)
2   {
3       $nomeTabela = $estrutura[0];
4       $variaveis = "";
5       for($i=0; $i<count($estrutura[1]); $i++)
6       {
7           if($i == 0)
8               $variaveis .= $estrutura[1][$i];
9           else
10              $variaveis .= ",".$estrutura[1][$i];
11      }
12      $sql = "Select $variaveis from $nomeTabela";
13      return $sql;
14  }
```

Figura 34 - Montar SQL

A função responsável por montar as consultas SQL que buscam os dados de cada tabela (Figura 34) recebe como parâmetro a estrutura de uma das tabelas da base que está sendo importada, em que na primeira posição do *array* é recuperado o nome da tabela que se quer selecionar os dados (Linha 3). As colunas são percorridas (Linha 5) e os parâmetros da SQL retornada são estabelecidos (Linhas 8 e 10), ao final da função é devolvido uma consulta que recupera todos os dados de uma tabela (Linha 13). A consulta montada através da função apresentada na Figura 34 é executada separadamente em outra função (Figura 35).

```
1  public static function ExecutarSQL($sql,$estruturaEspecifica,$estruturaGeral)
2  {
3      $db->setFetchMode(DB_FETCHMODE_ORDERED);
4      $consulta = $db->query($sql);
5      $resultado = array();
6      $aux = 0;
7      $campos = $estruturaEspecifica[1];
8      while($data = $consulta->fetchRow())
9      {
10         $array = array();
11         for($i=0;$i<count($campos);$i++)
12         {
13             if($campos[$i] != $estruturaEspecifica[2]['columnName'])
14             {
15                 $array[$campos[$i]] = utf8_encode($data[$i]);
16             }
17             else
18             {
19                 $sql = gerador::MontarSQL($estruturaGeral[$estruturaEspecifica[2]
20                                         ['referenceTableName']])." where ".$estruturaEspecifica[2]
21                                         ['referenceColumnName']." = ".$data[$i];
22                 $relacao = gerador::ExecutaSQL($sql,$estruturaGeral[$estruturaEspecifica[2]
23                                                 ['referenceTableName']],$estruturaGeral);
24                 $array[$estruturaEspecifica[2]['referenceTableName']] = $relacao[0];
25             }
26         }
27         $resultado[$aux] = $array;
28         $aux++;
29     }
30     return $resultado;
31 }
```

Figura 35 - Executar SQL

Na Linha 3 (Figura 35) é definido o formato de retorno que será aplicado para execução da consulta “DB_FETCHMODE_ORDERED” (retorna o resultado da consulta em formato de *array*), fazendo com que seja possível através de um lanço de repetição (Linha 11) montar dinamicamente o retorno independente do número de campos ou origem dos dados (exemplo de passagem pelo laço de repetição: “$array[‘nome’] = $data[0]”). O retorno da consulta executada (Linha 4) é uma lista contendo todos os dados registrados em uma tabela da base de dados relacional. Assim, cada registro retornado (que representa a instância de uma linha da tabela) é percorrido (Linha 8) e, em seguida, cada posição do *array* que representa uma coluna é preenchido com os valores de cada registro.

Vale ressaltar que o tratamento realizado na Linha 13 possibilita verifica se o campo retornado é uma relação com outra tabela (uma chave estrangeira). Caso seja, é chamada a função que monta SQL para retornar os registros de forma recursiva (Linha 19). Por exemplo, caso a função estivesse recebendo a estrutura de uma tabela “Pessoa”, que tem relação com “Endereco” e esta, por sua vez, tivesse relação com “Cidade”, recursivamente seria montado um *array* com os valores que compõe a cidade, esse *array* seria inserido no endereço, que, por fim, o *array* composto pelos dados de endereço seria relacionado como um dos campos do *array* de pessoa. Esse processo permite que o modelo relacional seja convertido para o

modelo orientado a documentos, fazendo com que o documento “Pessoa” seja composto pelo documento “Endereco” e este, por sua vez, composto pelo documento “Cidade”.

As funções apresentadas nessa seção são responsáveis por importar os dados de uma base relacional para o modelo orientado a documentos. Na seção seguinte será apresentada a construção dos filtros dinâmicos que consultam os *datasources* importados pelos usuários.

4.9.2 Construção Dinâmica dos Filtros

A criação das consultas, baseadas nas configurações estabelecidas pelo usuário, é centrada em três tabelas da base de dados da ferramenta: “Campo”, “Configuracao” e “Filtro”. A Figura 36 apresenta o processo de construção da consulta SQL que retorna os valores filtrados conforme as necessidades do usuário.

```
1  public static function ExecutarFiltro($idObjeto,$datasource)
2  {
3      $id = $idObjeto;
4      $sql = "select c.tabelaBase, c.nomereal, replace(co.descricao,'@valor',f.valor) as condicao,
5              f.valor from filtro f, campo c, condicao co where f.idobjeto = $id and f.idcampo = c.id
6              and f.idcondicao = co.id";
7      $filtros = $db->query($sql);
8      $m = new Mongo();
9      $dbm = $m->selectDB($datasource);
10     $array = array();
11     while($data = $filtros->fetchRow())
12     {
13         $collection = $dbm->selectCollection($data->tabelaBase);
14
15         if(strstr($data->condicao, '=')
16             $array[$data->nomereal] = $data->valor;
17         else if(strstr($data->condicao, 'not'))
18         {
19             $auxRetorno = explode(",",$data->condicao);
20             $array[$data->nomereal] =  array($auxRetorno[0] => new MongoRegex($auxRetorno[1]));
21         }
22         else if(strstr($data->condicao, '/'))
23             $array[$data->nomereal] = new MongoRegex($data->condicao);
24         else
25             $array[$data->nomereal] =  array($data->condicao => "".$data->valor);
26     }
27     $dados = $collection->find($array);
28     return $dados;
29 }
```

Figura 36 - Executar Filtro

A função apresentada na Figura 36 recebe como parâmetro o objeto que tem seus dados filtrados e o nome do *datasource* que os dados têm origem (Linha 1). Inicialmente, é realizada uma consulta que retorna a lista de filtros (Linha 4). Como exemplo de retorno, pode-se apresentar a seguinte sequência de valores (“Curso”,”nomeDoCurso”,”/^Sis/”), que poderia ser traduzido em linguagem natural como “Todos os cursos cujo nome inicie com Sis”. Em seguida cada filtro é percorrido (Linha 11) e, dependendo do tipo de instrução, sua

aplicação é realizada de forma diferenciada. Primeiramente, é verificado se é uma operação de igualdade (Linha 15), caso seja, é adicionado ao *array* de coleção de filtros seguindo o modelo: $array[“nomeDoCurso”] = “Sistemas de Informação”. Os tipos de tratamento, além de igualdade, podem ser: similar ao “*not like*” do SQL, ao “*like”* e a comparações como “maior que”, “menor que” etc..

A seção seguinte apresenta a forma com que foi implementado o padrão de projeto *observer* na ferramenta.

4.9.3 Aplicação do Padrão de Projeto *Observe*

A concepção do padrão de projeto *observer* para o protótipo se deu de forma diferenciada, pois os objetos dependentes não estão centralizados, já que os usuários que trabalham de forma colaborativa estão em estações de trabalho diferentes e necessitam manter suas interfaces coerentes com os demais usuários. A Figura 37 apresenta a função que é responsável por atualizar o estado da classe *Subject*.

```
1   public static function verificaVersao($versaoAtual)
2   {
3       $_SESSION['versao'] = $versaoAtual;
4       set_time_limit(0);
5       while (true)
6       {
7           sleep(1);
8           $sql = "SELECT Max(versao) as UltimaVersao
9               FROM VersaoProjeto vp
10              where vp.idProjeto = $idProjeto
11              group by versaoProjeto;
12
13          $ultimaVersao = $db->getOne($sql);
14
15          if($ultimaVersao != $_SESSION['versao'])
16          {
17              $dados = ControleDeVersao::buscarObjetosPorVersao($_SESSION['versao']);
18              echo '<script type="text/javascript">
19                  window.parent.alterarEstado($dados);
20            </script>';
21
22              ob_flush();
23              flush();
24          }
25  }
```

Figura 37 - Função Comet

A Figura 37 implementa um modelo denominado *comet*, que funciona como um Ajax reverso, em que atualizações são passadas para o navegador sem que o navegador solicite de forma explicita. Desta forma, assim que o usuário acessa um projeto de relatório, o *Subject*

chama uma função *php* denominada “verificaVersao” (Figura 37). Inicialmente esse método armazena em uma variável de sessão a versão atual do relatório que está sendo trabalhado (Linha 3). É iniciado um laço de repetição infinito (Linha 5), que tem como intenção executar operações como um serviço contínuo. Porém, para não sobrecarregar o servidor, foi definido o intervalo de 1 segundo entre uma consulta e outra (Linha 7). em seguida é realizada uma consulta na base de dados, que retorna a última versão de um projeto (Linha 13). Caso a versão retornada seja diferente da armazenada em sessão (Linha 15), será criada uma lista com os objetos que tiveram modificação devido à mudança de versão (podendo ser uma inserção, atualização ou deleção de objeto) (Linha 17). Por fim, uma instrução constrói uma chamada *javascript* que executa uma função passando como parâmetro a versão atual e a lista de objetos afetados (Linha 18). Através da função nativa do PHP ob_flush() essa instrução é imprimida no navegador sem que o laço de repetição seja interrompido. Na Figura 38 é apresentado o tratamento necessário para comunicar o *javascript* (contido no HTML) com o *actionscript* (que é utilizado no desenvolvimento com Flex).

1
```
1  <SCRIPT LANGUAGE="JavaScript">
2      function alterarEstado($objetos,$ultimaVersao) {
3          idFlex.funcaoFLex($objetos,$ultimaVersao);
4      }
5  </SCRIPT>
```

2
```
1  <mx:Script>
2      import flash.external.*;
3      public function iniciar():void {
4         ExternalInterface.addCallback("funcaoFlex",dispararNotificacao);
5      }
6
7      public function dispararNotificacao(objetos:ArrayCollection,ultimaVersao:int):void {
8         subject.SetStatus(objetos,ultimaVersao);
9      }
10 </mx:Script>
```

Figura 38 - Integração entre o javascript e o actionscript

A função comet ao chamar uma função do *javascript*, torna necessário que o *javascript* passe os valores de atualização para o *Flex*. Essa integração entre as duas linguagens de *script* é apresentada na Figura 38. O Item 1 apresenta a função *javascript* chamado pelo comet, em que através do identificador do *Flex* dentro do navegador chama a função que irá receber seus parâmetros (Linha 3). O *Flex*, ao carregar a página pela primeira vez, declara um evento dinâmico responsável por chamar a função que se comunica com

Subject (Linha 4), a qualquer momento que o evento é disparado ele chama a função "dispararNotificacao" (Linha 7) que aciona o *Subject* e atualiza seu estado. A função que é responsável por notificar os observadores é apresentada a seguir (Figura 39).

```
1  public function SetStatus(objetos:ArrayCollection, versao:int):void
2  {
3      versaoAtual = versao;
4      Notify(objetos);
5  }
6
7  public function Notify(objetos:ArrayCollection, versao:int):void
8  {
9      foreach(listaObservers as obj)
10     {
11         foreach(objetos as o)
12         {
13             if(obj.id == o.id)
14                 obj.Update(o);
15         }
16     }
17 }
```

Figura 39 - Função de Notificação

Ao ser inserido um novo estado, o *Subject* guarda a nova versão e dispara a função *Notify()* (Linha 4). A função *Notify()* por sua vez percorre sua lista de observadores (Linha 9) e a cada observador verifica se sofreu alguma alteração na mudança de versão (Linha 13), caso sim é disparado a função de atualização (Linha 14). A seguir é apresentada a função de atualização de um observador.

```
1  public function Update(objeto:Object):void
2  {
3      versao = Subject.getEstado();
4      if(objeto.acao == "deletar")
5          Remover(objeto);
6      else if(objeto.acao == "atualizar")
7              Atualizar(objeto);
8      else
9          Inserir(objeto);
10 }
```

Figura 40 - Função de atualização do observador

A função de atualização do observador (Figura 40) verifica o tipo de ação (Linhas 4 e 6) e conforme a instrução pode ser realizada a deleção, atualização ou inserção do objeto. A

Figura 41 apresenta um diagrama de sequência que resume os passos realizados pela aplicação na aplicação do padrão *observer* no protótipo.

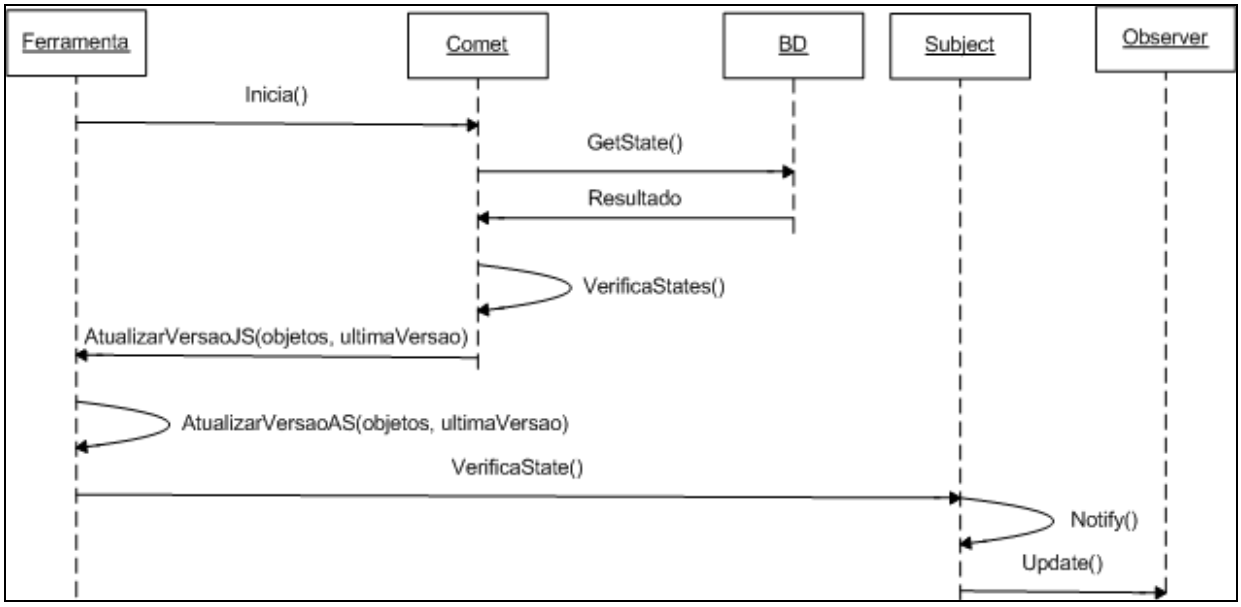


Figura 41 - Diagrama de Sequência da Aplicação do Padrão de Projeto *Observer* no Protótipo

Pode-se notar no diagrama de sequência apresentado na Figura 41 que sua composição resume todos os passos de aplicação do padrão *observer* descritos nesta seção. Inicialmente após o usuário acessar um projeto de relatório na ferramenta é iniciada a função Comet, que em intervalos de tempo acessa o banco de dados relacional para verificar o estado atual do relatório que está sendo trabalhado. Caso tenha uma mudança de estado é enviado através do *javascript* o novo estado do relatório, a função chamada no *javascript* se comunica com o *actionscript* enviando as informações. Desta forma, o *actionscript* pode enviar o novo estado para o *Subject* que irá notificar seus observadores.

A seção a seguir apresenta as considerações sobre os resultados e discussões do presente trabalho.

4.10 CONSIDERAÇÕES

No decorrer da seção de resultados e discussões foram descritos os artefatos produzidos no desenvolvimento do protótipo da ferramenta. As soluções apresentadas estão centradas no conhecimento adquirido dos conceitos discorridos na revisão de literatura (sessão 2), que são: Sistemas Colaborativos, o Padrão de Projeto *Observer* e a classe de banco de dados NOSQL.

A aplicação do Modelo 3C foi focada no ambiente de cooperação, em que o controle de objetos manipulados de um relatório é realizado com auxílio do Padrão de Projeto *Observer*. O Padrão de Projeto *Observer* aplicado para ferramenta teve que ser trabalhado de forma diferenciada, pois para que o *Subject* tivesse seu estado atualizado foi necessário criar uma função do lado do servidor que executa uma operação de consulta de estado como um serviço contínuo. Além disso, optou-se por um controle maior do *Subject* sobre os observadores, pois além de seu novo estado, ele também recebe os objetos que sofreram modificação, isso permite que a notificação seja disparada apenas para os observadores de interesse.

A utilização da classe de banco de dados NOSQL foi definida como solução no que tange a flexibilidade da ferramenta de aceitar diversas estruturas de dados e oferecer características de alta escalabilidade e disponibilidade. A utilização do banco de dados MongoDB contribuiu para o desenvolvimento do trabalho, pois os parâmetros das funções oferecidas pelo MongoDB (para acessar bancos de dados, localizar tipos de documento e fazer consultas) são passadas via *String*, o que permitiu construir a manipulação dos dados de forma dinâmica.

Na próxima seção são apresentadas as considerações finais do presente trabalho.

5 CONSIDERAÇÕES FINAIS

O presente trabalho teve por objetivo desenvolver um protótipo de ferramenta, que tem por finalidade permitir que através de soluções colaborativas a ferramenta sirva de apoio à geração de relatórios, disponibilizando recursos para coordenação, comunicação e cooperação, além de oferecer flexibilidade no que tange a possibilidade do usuário indicar diversas estruturas de dados para trabalhar. Os relatórios elaborados são preenchidos com dados consultados dinamicamente através do retorno das consultas dos *datasources* advindos da importação de bases de dados relacionais. Considerando os objetivos definidos, o propósito geral do trabalho consistiu na demonstração da possibilidade de utilizar o Modelo 3C de Colaboração, o Padrão de Projeto *Observer* e a classe de banco de dados NOSQL como soluções para problemas encontrados no desenvolvimento do protótipo do sistema colaborativo proposto neste trabalho.

No desenvolvimento do trabalho foi considerado que um sistema colaborativo deve oferecer comunicação, coordenação e cooperação, assim, o Modelo 3C de colaboração se mostrou eficiente, pois sua abordagem relaciona essas três dimensões e define a forma de interação entre elas. Através do Modelo 3C foi possível definir os *groupwares* necessários para o desenvolvimento do protótipo. Porém, deve-se considerar que o trabalho se trata de um protótipo, cujo objetivo principal é demonstrar a utilização da classe de banco de dados NOSQL como solução de escalabilidade, disponibilidade e flexibilidade para aceitar qualquer tipo de estrutura de dados, além do Padrão de Projeto *observer* para gerenciar os objetos do ambiente de cooperação que apresentam dependências entre si. Logo, referente ao Modelo 3C de colaboração, apesar do protótipo ter *groupwares* referentes às três dimensões do modelo, o foco foi centrado na dimensão da cooperação, inserindo elementos de percepção para controle e auxílio do trabalho colaborativo.

O Padrão de Projeto *Observer* aplicado no protótipo se mostrou como uma boa opção para controlar as dependências entre os objetos de um mesmo relatório. O ambiente de cooperação previsto no Modelo 3C se deu de forma mais organizada, já que a função de verificar as alterações em cada ambiente ficou centrada em uma única classe *Subject*. Além disso, o padrão deixou a ferramenta preparada para futuras atualizações e agregação de novas ferramentas. Isso porque o controle de dependências fica centrado no *Subject*, assim cada nova funcionalidade só precisará inserir seu observador na lista de observadores do *Subject*, além de implementar sua função de atualização *Update()*.

A classe de banco de dados NOSQL se apresentou como uma boa solução para resolução de possíveis problemas de escalabilidade (por não se ter a informação do volume de dados que o usuário pode importar) e flexibilidade no que tange um dos objetivos específicos do trabalho, que é permitir que o usuário importe qualquer tipo de estrutura de dados. Dentre os principais tipos de banco de dados NOSQL, para o protótipo proposto neste trabalho, foi escolhido o banco de dados orientado a documentos, pois não apresenta uma estrutura pré-definida, podendo ser alterada a qualquer momento, além de oferecer alta escalabilidade e disponibilidade. O banco de dados orientado a documentos escolhido foi o MongoDB que oferece vários dos recursos avançados de consulta (como filtragem, agregação e classificação), que foram necessários para a construção dos filtros dinâmicos previstos para ferramenta. Ainda o MongoDB já oferece soluções de *sharing* de forma nativa possibilitando que se crie *clusters* do banco de dados, garantindo alta escalabilidade com a inserção de novos nós a *grid*.

Sabe-se que o banco de dados orientado a documentos não garante a consistência dos dados, entretanto, como a ferramenta não permite que os dados sejam alterados depois de importados, isso não se mostrou como um problema. Além disso, a importação dos dados poderia ser feita via *script* de importação de banco de dados, mas seria uma solução mais complexa. A solução apresentada recebendo via formulário os dados de acesso à base relacional é suficiente para demonstrar a classe de banco de dados NOSQL como solução.

Sobre as considerações gerais do protótipo deve-se destacar que ele oferece as possibilidades de recuperar ou inserir novos relatórios. Quando os relatórios são editados pelo usuário a reconstrução é realizada através da recuperação dos atributos de cada componente do relatório. Estes atributos são armazenados em uma base de dados utilizada pela ferramenta. Desta forma, o gestor tem a possibilidade de definir relatórios para utilizar periodicamente e adaptá-los conforme a necessidade de cada período. A partir da experiência adquirida no desenvolvimento da ferramenta, com a utilização do *Framework Adobe Flex*, foi possível observar grandes vantagens em relação ao *Framework*, principalmente, quando relacionado à ambientes gráficos, pois apresenta recursos nativos de comunicação assíncrona e manipulação de componentes gráficos. Ainda contempla uma comunidade de desenvolvedores que alimenta as possibilidades de sua utilização, com recursos não nativos.

Este trabalho abre a possibilidade para diferentes formas de continuação. Dentre elas podem ser citadas:

- Realizar um estudo comparativo entre banco de dados orientado a documentos, destacando características de performance, segurança, *clustering* etc.;

- Fazer o levantamento dos *groupwares* necessários para a ferramenta proposta neste trabalho, pois o presente trabalho abordou apenas alguns, suficientes para atingir o objetivo principal do trabalho.
- Incorporar a ferramenta à plataforma Node.Js, que é construída sobre *JavaScript* para desenvolver aplicações leves e escaláveis (NODEJS, 2012). O que seria muito útil para os ambientes de cooperação que necessitam atualizar as informações do ambiente quase que instantaneamente, para não prejudicar o trabalho colaborativo.

6 REFERÊNCIAS BIBLIOGRÁFICAS

ABADI, Daniel J. **Query Execution in Column-Oriented Database Systems.** 2008. 148 p. Thesis (Doctor of Philosophy in Computer Science and Engineering) – Massachusetts Institute of Technology, Massachusetts, USA.

ALMEIDA, Ricardo Cardoso de. **Desenvolvimento de uma Ferramenta Gráfica para Customização de Relatórios a Partir de Dados Extraídos do Google Analytics.** 2011. 39 p. Trabalho de Estágio (Curso de Sistemas de Informação) CEULP/Ulbra, Palmas, Tocantins.

ASSIS, Rodrigo Lemos de. **Facilitando a Percepção em Ambientes Virtuais de Aprendizado Através da Abordagem *Groupware*.** 2000. 155 p. Dissertação (Mestrado em Ciências em Informática) – Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro.

BORGES, Robson M.; PINTO, Sérgio Crespo C. S.; BARBOSA, Jorge L. V.; BARBOSA, Débora N. F.. Usando o Modelo 3C de Colaboração e Vygotsky no Ensino de Programação Distribuída em Pares. In: SIMPÓSIO BRASILEIRO DE INFORMATICA NA EDUCAÇÃO, 18, 2007, São Paulo. ***Anais...*** São Paulo, 2007.

BRITO, Ricardo W. Bancos de Dados **NoSQL x SGBDs Relacionais:** Análise Comparativa. In: InfoBrasil, 3, 2010, Fortaleza. ***Anais...*** Fortaleza, 2010.

CAMARA, Evandro Augusto Muchinski; GARCIA, Renato Joukoski. **Avaliação de Sistemas de Reconhecimento de Fala Para Indexação Automática De Vídeos.** 2011. 39 p. Monografia (Bacharel em Ciências da Computação) – Universidade Federal do Paraná, Curitiba, Paraná.

COELHO, Sérgio Salles; NOVAES, Celso Carlos. Modelagem de Informações para Construção (BIM) e Ambientes Colaborativos para Gestão de Projetos na Construção Civil. In: WORKSHOP BRASILEIRO GESTÃO DO PROCESSODE PROJETOS NA CONSTRUÇÃO DE EDIFÍCIOS, 8, 2008, São Paulo. ***Anais...*** São Paulo, 2008.

DIANA, Mauricio De; GEROSA, Marco Aurélio. **NOSQL na Web 2.0:** Um Estudo Comparativo de Bancos Não-Relacionais para Armazenamento de Dados na Web 2.0. In: Workshop de Teses e Dissertações em Banco de Dados, 9, 2010, Belo Horizonte. ***Ainais...*** Belo Horizonte, 2010.

ELLIS, G. A.; GIBBS, S. J.; REIN, G. L. Groupware: Some Issues and Experiences. **Magazine Communications of the ACM.** New York, v. 34, p. 39-58, janeiro de 1991.

FILIPPO, Denise Del Re. **Suporte à Coordenação em Sistemas Colaborativos:** Uma Pesquisa-Ação com Aprendizes e Mediadores Atuando em Fóruns de Discussão de Um Curso a Distância. 2008. 282 p. Tese (Doutorado em Informática) – Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro.

FUKS, Hugo; RAPOSO, Alberto Barbosa; GEROSA, Marco Aurélio; LUCENA, Carlos José Pereira de. **O Modelo de Colaboração 3C e a Engenharia de Groupware.** In: Ned Kock (Org.) **Encyclopedia of E-Collaboration.** New York: Hershey, 2008, p. 637-644.

GAMMA, Erich; HELM, Richard; JOHNSON, Ralph; VLISSIDES, John. **Padrões de Projeto:**Soluções Reutilizáveis de Software Orientado a Objetos. Tradutor SALGADO, Luiz A. Meirelles. Porto Alegre: Bookman, 2000. 364 p.

GEROSA, Marco Aurélio. **Desenvolvimento de Groupware Componentizado com Base no Modelo 3C de Colaboração.** 2006. 276 p. Tese (Doutorado em Informática) – Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro.

LEITE, Gleidson Sobreira. **Análise Comparativa do Teorema CAP Entre Bancos de Dados NoSQL e Bancos de Dados Relacionais.** 2010. 49 p. Monografia (Bacharel em Ciências da Computação) – Faculdade Farias Brito, Fortaleza.

LÓSCIO, Bernadette Farias; OLIVEIRA, Hélio Rodrigues de; PONTES, Jonas César de Sousa. NoSQL no desenvolvimento de aplicações Web colaborativas. In: SIMPÓSIO BRASILEIRO DE SISTEMAS COLABORATIVOS, 8, 2011, Paraty. ***Anais...*** Paraty: SBC, 2007.

MATTEUSSI, Kassiano José. Protótipo **De Interface Web Com Php Para Gerenciamento de Banco de Dados CouchDB.** 2010. 81 p. Monografia (Bacharel em Ciências da Computação) – Universidade Comunitária Da Região De Chapecó, Chapecó, Santa Catarina.

NODEJS. Disponível em: http://www.nodejs.org. Acesso em: 05 de junho de 2012

OLIVEIRA, Eduardo Araújo; TEDESCO, Patrícia. *i-collaboration*: Um Modelo de Colaboração Inteligente Personalizada para Ambientes de EAD. **Revista Brasileira de Informática na Educação.** v. 18, p. 17-31, 2010.

PIMENTEL, Mariano; FUKS, Hugo. **Sistemas Colaborativos.** Rio de Janeiro: Elsevier, 2011. 375 p.

SANTOS, Nuno Miguel Queirós Arantes dos. **Bases de Dados alternativas para Websites.** 2011. 130 p. Dissertação (Mestrado em Engenharia Informática e Computação) – Faculdade De Engenharia Da Universidade Do Porto, Porto, Portugal.

SHAO, Bin; WANG, Haixun; XIAO, Yanghua. **Managing and Mining Large Graphs:** Systems and Implementations. In: ACM International Conference on Management of Data (SIGMOD), 2012, Arizona. ***Anais...*** Arizona: ACM, 2012.

SICA, Fernando Cortez. **Camada Groupware:** uma camada de suporte a aplicações cooperativas em ambientes distribuídos abertos. 1996. 171 p. Dissertação (Mestrado em Engenharia Elétrica) – Universidade Estadual de Campinas, Campinas, São Paulo.

SILVA, Patrícia Freitas da; PENHA, José Afonso Moraes; ALVES, Gabriel Marcelino. Estudo do Padrão de Projeto *Observer* no Desenvolvimento de Softwares Utilizando a Arquitetura MVC. In: MOSTRA NACIONAL DE INICIAÇÃO CIENTÍFICA E TECNOLÓGICA INTERDISCIPLINAR, 3, 2009, Camboriú. ***Anais...*** Camboriú: Universidade Federal de Santa Catarina, 2009.

SILVA, Tiago Pasqualini da. **Cassandra –** Uma sistema de armazenamento NoSQL altamente escalável. Disponível em: http://www2.sor.ufscar.br/verdi/topicosCloud/Cassandra.pdf. Acesso em: 22 de maio de 2012.

STRAUCH, Christof. **NoSQL Databases.** Disponível em: http://www.christof-strauch.de/nosqldbs.pdf. Acesso em: 25 de abril de 2012.

TAIT, Tania Fatima Calvi; SILVEIRA, Mário César; SIMÕES, Rafael; CYBIS, Walter. Desenvolvimento de Software para Trabalho Cooperativo Auxiliado por Computador: Uma Abordagem Ergonômica. **Revista UNIMAR.** Maringá, v. 19, n. 4, p. 969-995, 1997.

TIETZE, Daniel A. **A Framework for Developing Component-based Co-operative Applications.** 2001. 173 p. Dissertation (Doktor-Ingenieurs) – Technischen Universität Darmstadt, Darmstadt, Germany.

TIRELO, Fabio; BIGONHA, Roberto da Silva; BIGONHA, Mariza Andrade da Silva; VALENTE, Marco Túlio de Oliveira. **Desenvolvimento de Software Orientado por**

Aspectos. In: JORNADA DE ATUALIZAÇÃO EM INFORMÁTICA, 23, 2004, Salvador. ***Anais...*** Salvador: SBC, 2004.

TORRES, Patrícia Lupion; ALCANTARA, Paulo R.; IRALA, Esrom Adriano Freitas. Grupos de Consenso: Uma Proposta de Aprendizagem Colaborativa para o Processo de Ensino-Aprendizagem. **Revista Diálogo Educacional.** Paraná, v. 4, n. 13, p. 1-17, setembro a dezembro de 2004.

TOTH, Renato Molina. **Abordagem NoSQL –** uma real alternativa. Disponível em: http://www2.sor.ufscar.br/~verdi/topicosCloud/nosql_artigo.pdf. Acesso em: 16 de maio de 2012.

WINOGRAD, Terry. A Language/Action Perspective on the Design of Cooperative Work. In: **Human-Computer Interaction**, 3:1, 1987-88, p. 3-30.

ZOTTO, Ozir Francisco De Andrade. **Um Estudo dos Efeitos Organizacionais e Sociais da Utilização da Tecnologia Groupware Lotus Notes na Administração Pública do Estado do Paraná.** 2000. 176 p. Dissertação (Mestrado em Ciência da Computação) – Universidade Federal do Rio Grande do Sul, Porto Alegre, Rio Grande do sul.